The 35th Conference on Computational Linguistics and Speech Processing (ROCLING 2023)
Taipei City, Taiwan, October 20-21, 2023. The Association for Computational Linguistics and Chinese Language Processing

Sentence-level Revision with Neural Reinforcement Learning

Zhendong Du
Waseda University
Kitakyushu, Japan
zhendong@fuji.waseda.jp

Kenji Hashimoto
Waseda University
Kitakyushu, Japan
kenji.hashimoto@waseda.jp

Abstract

The objective of Sentence-level Revision (SentRev) is to enhance the fluency of English writing; however, the performance of the three baseline methods is notably suboptimal. In this study, we propose a method utilizing neural reinforcement learning, tailored to the specific characteristics of this task, which has resulted in superior performance over the baseline methods, surpassing them in multiple evaluation metrics. Moreover, we have identified conspicuous bottlenecks in SentRev's efficacy in improving the fluency of English writing.

Keywords: English Writing Assistant, SentRev, NRL

1 Introduction

The inadequate English writing proficiency of many non-native English speakers renders their academic English writing a challenging task, hence academic writing assistant has become a popular downstream task in the field of Natural Language Processing (NLP). However, much of the previous work has predominantly concentrated on English Grammatical correction (GEC), with scarce results published concerning the more challenging aspect of English writing fluency enhancement.

(Ito et al., 2019) has introduced Sentence-level-revision (SentRev), a task dedicated to enhancing the fluency of English writing. The authors have established baseline performance for the task at hand by employing methodologies from a variety of other Natural Language Processing tasks. However, significant room for improvement in baseline performance remains. In pursuit of an optimized approach for the task at hand, we conducted a comprehensive analysis of its characteristics. Our investigation revealed that the task inherently involves iterative sentence-level revisions aimed at enhancing English writing fluency. This aspect aligns closely with the self-improving nature of reinforcement learning, which continuously refines its performance to achieve superior outcomes. Consequently, we adopted a reinforcement learning paradigm tailored to the unique requirements of this task and employed the GLUE (Wang et al., 2018) as the reward function to drive the optimization process. An evaluation was conducted on the SMITH dataset (Ito et al., 2019), and the results substantiated that our proposed method exhibits a significant improvement over the baseline performance. Additionally, our experimental findings have revealed limitations within SentRev, resulting in conspicuous bottlenecks in the enhancement of English writing fluency.

2 Related Works

2.1 Grammatical Error Correction (GEC)

The objective of Grammatical Error Correction (GEC) is to transform a sentence S with grammatical errors into a corrected version, denoted as S'. Given its nature of transforming a sequence output into a new sequence, modern approaches to this task commonly treat it as a machine translation problem. In essence, it involves ″translating″ a sentence with grammatical errors into a corrected sentence.

With the introduction of the Transformer (Vaswani et al., 2017), significant advancements have been made in the GEC task over the past few years, particularly in the do-

main of English Grammatical Error Correction (Yuan and Briscoe, 2016; Omelianchuk et al., 2020; Stahlberg and Kumar, 2020). Most grammatical errors in English can now be effectively rectified. However, for non-native English speakers, improving the fluency of their English writing poses a greater challenge, especially when engaging in academic English writing, as non-fluent English expression may hinder their ability to effectively present their academic viewpoints. Unfortunately, the enhancement of English writing fluency has received limited attention in research due to its entanglement with numerous linguistic intricacies.

2.2 Sentence-level Revsion (SentRev)

(Ito et al., 2019) proposes Sentence-level Revision (SentRev) to address the challenge of improving English writing fluency. This task conceptualizes the enhancement of English fluency as the act of rewriting sentences. The specific process is illustrated in Figure 1.

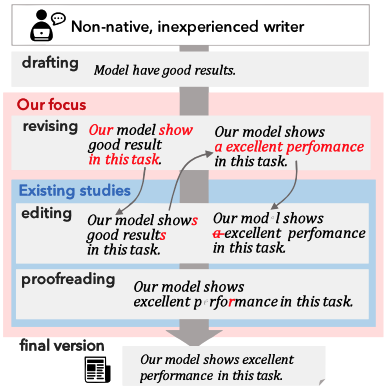

Figure 1: Overview of the process of SentRev. Figure copied from (Ito et al., 2019)

In this endeavor, the authors have constructed a manually annotated test dataset named the SMITH dataset for evaluating SentRev. Subsequently, three distinct NLP downstream task models were employed for this purpose, namely, the Heuristic noising and denoising model, the Enc-Dec noising and denoising model, and the GEC model (Zhao et al., 2019). These models were used to establish baseline scores on the SMITH dataset, however, the attained baseline scores were deemed unsatisfactory.

2.3 Neural Reinforcement Learning (NRL)

Neural Reinforcement Learning (NRL) is a synthesis of Reinforcement Learning (RL) and Deep Learning, leveraging the expressive power of neural networks to approximate complex functions that represent the state and action spaces (Mnih et al., 2015).

In traditional RL, an agent learns to take actions in an environment to maximize some notion of cumulative reward. The learning process is often guided by the Bellman equation:

$$V(s) = \max_a \left(R(s,a) + \gamma \sum_{s'} P(s'|s,a)V(s') \right) \quad (1)$$

In NRL, deep neural networks are utilized to approximate the value functions $V(s)$ or the policy $\pi(a|s)$, allowing the approach to handle high-dimensional state and action spaces (Gu et al., 2016).

In the context of Natural Language Processing (NLP) downstream tasks, RL has been employed in various applications including MT (Wu et al., 2018), GEC (Sakaguchi et al., 2017), and Text Style Transfer (Gong et al., 2019), achieving amazing performance. The resemblances between these downstream tasks and SentRev provide valuable insights and precedents for the application of RL in SentRev.

3 Proposed Method

In order to address the issue of low compatibility between the baseline method and SentRev, We propose a method based on NRL for SentRev. This proposition emerges from our observation that the rewriting process of SentRev is congruent with the characteristics inherent to NRL. We have engineered a comprehensive NRL method specifically tailored for SentRev. The aim of this method is to maximize the expected GLEU score through the optimization of model parameters, and it has been specifically adjusted in accordance with the task requirements. All these components and choices collectively delineate a complete, explicit, and coherent method that can be employed for the transformation of non-fluent drafts into fluent English sentences adhering to an academic style. The high-level

description of the training procedure is shown in Algorithm 1. Details of the specific design are delineated below.

3.1 State Representation

Assume that the source sentence (referred to as the Draft) is denoted by $S = \{s_1, s_2, ..., s_m\}$, the currently rewritten portion is denoted by $H = \{h_1, h_2, ..., h_t\}$, and the academically fluent English sentence (referred to as the Reference) is denoted by $R = \{r_1, r_2, ..., r_n\}$.

Word Embedding Representation By utilizing Word2Vec (Mikolov et al., 2013), each word is mapped into a K-dimensional space.

$$\mathbf{S} = \text{Embed}(s_i) \in \mathbb{R}^{m \times K} \tag{2}$$

$$\mathbf{H} = \text{Embed}(h_i) \in \mathbb{R}^{t \times K} \tag{3}$$

$$\mathbf{R} = \text{Embed}(r_i) \in \mathbb{R}^{n \times K} \tag{4}$$

Position Encoding Position encoding is introduced to capture sequential information within the sentence.

$$\mathbf{S}_{\text{pos}} = \text{PosEncode}(\mathbf{S}) \in \mathbb{R}^{m \times K} \tag{5}$$

$$\mathbf{H}_{\text{pos}} = \text{PosEncode}(\mathbf{H}) \in \mathbb{R}^{t \times K} \tag{6}$$

$$\mathbf{R}_{\text{pos}} = \text{PosEncode}(\mathbf{R}) \in \mathbb{R}^{n \times K} \tag{7}$$

Length Information The length of the sentence can be represented as a scalar feature.

$$\mathbf{L}_S = m \tag{8}$$

$$\mathbf{L}_H = t \tag{9}$$

$$\mathbf{L}_R = n \tag{10}$$

N-gram Overlap Compute the n-gram overlap statistics between H and S, and H and R, and subsequently normalize them.

$$\mathbf{O}_{HS} = \frac{\text{Overlap}(H, S)}{\max(\text{Overlap}(H, S), \text{Overlap}(H, R))} \tag{11}$$

$$\mathbf{O}_{HR} = \frac{\text{Overlap}(H, R)}{\max(\text{Overlap}(H, S), \text{Overlap}(H, R))} \tag{12}$$

Final State Representation Concatenate the above features to form the final state representation.

$$\textbf{State} = \text{Concat}(\mathbf{S}_{\text{pos}}, \mathbf{H}_{\text{pos}}, \mathbf{R}_{\text{pos}}, \mathbf{L}_S, \mathbf{L}_H, \mathbf{L}_R, \mathbf{O}_{HS}, \mathbf{O}_{HR}) \tag{13}$$

Herein, Concat refers to the concatenation operation, and the ultimate **State** is the input to the model, encapsulating the current rewriting status, information pertaining to the source and reference sentences, as well as features related to length and n-gram overlap.

3.2 Strategy Network

The Strategy Network is tasked with generating the subsequent action based on the current state representation (e.g., selecting the next word). Below are the components and detailed equations of the Strategy Network.

Input Layer The input for the Strategy Network is represented by the state vector **State**.

Multi-Layer LSTM A sequence of LSTM (Hochreiter and Schmidhuber, 1997) layers is employed to capture potential long-range dependencies that might exist.

$$\mathbf{H}_1 = \text{LSTM}_1(\textbf{State}) \tag{14}$$

$$\mathbf{H}_2 = \text{LSTM}_2(\mathbf{H}_1) \tag{15}$$

$$\vdots \tag{16}$$

$$\mathbf{H}_L = \text{LSTM}_L(\mathbf{H}_{L-1}) \tag{17}$$

Here, $\mathbf{H}_i$ denotes the hidden state of the i-th layer, and L refers to the number of LSTM layers.

Output Layer The output layer transforms the output of the final LSTM layer into a probability distribution over the action space. Assuming that there are V possible actions (e.g., words in the vocabulary), the output layer can be defined as

$$\mathbf{P} = \text{Softmax}(\mathbf{W}\mathbf{H}_L + \mathbf{b}) \tag{18}$$

where $\mathbf{W} \in \mathbb{R}^{V \times D}$ and $\mathbf{b} \in \mathbb{R}^V$ are the parameters to be learned, and D represents the dimensionality of the output from the last LSTM layer.

Action Selection Finally, the action (e.g., the next word) is sampled from the probability distribution $\mathbf{P}$. Techniques such as temperature scaling can be employed to control the degree of randomness.

$$a_t = \text{Sample}(\mathbf{P}) \tag{19}$$

3.3 Value Function Network

The Value Function Network is devised to estimate the expected returns for a given state. This section delineates the key components and underlying mathematical formulations of the Value Function Network.

Input Layer The input for the Value Function Network is analogous to that of the Policy Network, both encompassing the state representation **State**.

Hidden Layers Multiple hidden layers are employed to capture the intricate representation of the state. The mathematical expressions for these layers can be presented as follows:

$$\mathbf{F}_1 = \text{ReLU}(\mathbf{W}_1\mathbf{State} + \mathbf{b}_1) \tag{20}$$
$$\mathbf{F}_2 = \text{ReLU}(\mathbf{W}_2\mathbf{F}_1 + \mathbf{b}_2) \tag{21}$$
$$\vdots \tag{22}$$
$$\mathbf{F}_H = \text{ReLU}(\mathbf{W}_H\mathbf{F}_{H-1} + \mathbf{b}_H) \tag{23}$$

In this framework, $\mathbf{F}_i$ denotes the activation of the i-th hidden layer, while $\mathbf{W}_i$ and $\mathbf{b}_i$ symbolize the corresponding weights and biases, respectively. H signifies the number of hidden layers.

Output Layer The output layer is constituted as a scalar, expressing the current state's value estimation:

$$V(\mathbf{State}) = \mathbf{W}_o\mathbf{F}_H + b_o \tag{24}$$

In this context, $\mathbf{W}_o$ and b_o denote the weights and biases of the output layer.

Training The training objective of the Value Function Network is to minimize the mean squared error between the estimated values and the actual returns. Let $\hat{V}(\mathbf{State})$ be the network's output and R be the actual return; the loss function is defined as:

$$\mathcal{L} = \frac{1}{N}\sum_{i=1}^{N}(\hat{V}(\mathbf{State}_i) - R_i)^2 \tag{25}$$

where N represents the number of samples.

3.4 Reward Function

The reward function delineates the methodology for assessing the value of each action based on the similarity between the model-generated output and the target reference output. In the context of this task, the reward function utilizes GLEU to gauge the resemblance between non-fluent English sentences, denoted as H, and the fluent English sentences in academic style, symbolized as R, with consideration of the n-grams in the source sentence S.

Computation of GLEU The GLEU score represents an automated evaluation metric, with the computation formula defined as:

$$\text{GLEU} = \min\left(1, \frac{|H|}{|S|}\right) \times \left(\frac{\sum_{n=1}^{N}\text{CountClipped}(n)}{\sum_{n=1}^{N}\text{Count}(n)}\right) \tag{26}$$

Wherein: $|H|$ and $|S|$ correspond to the lengths of sentences H and S, respectively. CountClipped(n) denotes the count of n-grams in H, with overlapping n-grams clipped to match the quantity present in R. Count(n) refers to the count of n-grams in H, without regard to the overlap with S. N signifies the maximum n-gram length under consideration.

Definition of Reward The reward function is characterized as the difference between the GLEU scores of the sentence produced by the current action and the preceding action:

$$\text{Reward} = \text{GLEU}(H_{\text{current}}, R, S) - \text{GLEU}(H_{\text{previous}}, R, S) \tag{27}$$

Such a definition of reward incentivizes the model to generate actions that augment the GLEU score.

Conclusion The reward function incentivizes the model by computing the GLEU score, thereby motivating the model to enhance the similarity with the reference sentence R, while simultaneously maintaining minimal overlap with the source sentence S. This function, in conjunction with the policy network and value function network, is utilized to train the NRL model, thus facilitating the learning of optimal parameters to maximize the expected GLEU score.

This design assures an optimal balance between academic style and fluency, by exclusively rewarding overlap with the reference sentence R, while concurrently penalizing unnecessary overlap with the source sentence S.

3.5 Algorithm Training

To maximize the anticipated GLEU score, we have opted for the following specific training algorithms and components:

Sampling Strategy We employ the epsilon-greedy strategy for balancing between exploration and exploitation. Specifically, a random action is chosen with a probability of ϵ, while an action recommended by the policy network is selected with a probability of $1-\epsilon$.

$$a_t = \begin{cases} \text{Random action} & \text{with probability } \epsilon \\ \text{Sample}(\mathbf{P}) & \text{with probability } 1-\epsilon \end{cases} \tag{28}$$

Optimization Algorithm Proximal Policy Optimization (PPO) (Schulman et al., 2017) is utilized as the primary optimization algorithm. PPO constrains the magnitude of policy updates by employing a clipped objective function.

$$\mathcal{L}_{\text{PPO}}(\theta) = \frac{1}{N}\sum_{i=1}^{N} \min\left(\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}A_i, \text{clip}\left(\frac{\pi_\theta(a_i|s_i)}{\pi_{\theta_{\text{old}}}(a_i|s_i)}, 1-\epsilon, 1+\epsilon\right)A_i\right) \tag{29}$$

Herein, π_θ is the current policy, $\pi_{\theta_{\text{old}}}$ is the policy prior to updating, and A_i is the advantage function.

Experience Replay We employ an experience replay buffer to store transitions and train the network through mini-batch random sampling.

The choices made in these configurations align with our objective of maximizing the expected GLEU score, reflecting a well-considered approach to the training process.

4 Experiments

Baseline Our baseline framework consists of three distinct models employed by (Ito et al., 2019), namely: the Heuristic Noising and Denoising Model (H-ND), the Encoder-Decoder Noising and Denoising Model (ED-ND), and the GEC model. Specifically, for the Noising and Denoising approach, the authors opted to select several sentences from the ACL Anthology Sentence Corpus (AASC)[1] and implemented a sequence of genetic rules to introduce noise directly into the dataset, thereby generating training material. Subsequently, the authors trained a denoising model utilizing the Transformer architecture as found in the fairseq (Ott et al., 2019) framework. In the case of the Encoder-Decoder Noising and Denoising approach, the authors employed three neural Encoder-Decoder structures to synthesize training data. These data, in conjunction with the datasets generated via the previously mentioned genetic methodology, were used to train the denoising model. Notably, the model architecture was identical to that of the heuristic model. Lastly, a pre-trained GEC model (Zhao et al., 2019) was harnessed as the third baseline model in the authors' investigative framework."

Data In the context of training the NRL model, we have utilized synthetic data generated within the baseline, serving as our source of training information. While the quality of these synthesized datasets may not compare favorably with the manually curated SMITH dataset, they represent the optimal choice for our purposes at this current juncture.

Hyperparameters The hyperparameters for the NRL model are shown in table 1:

Evaluation We evaluated our model using the SMITH dataset. The SMITH dataset consists of 10,000 pairs of data, divided equally into 5,000 pairs for the development set and 5,000 pairs for the test set. The underlying

[1] https://github.com/KMCS-NII/AASC

Algorithm 1 Sentence-level Revision with Neural Reinforcement Learning

Initialize: Actor network with parameters θ, Critic network with parameters ϕ, experience replay buffer $\mathcal{D}$
for epoch $= 1, \ldots,$ epochs **do**
  // Sampling from the experience replay buffer
  Sample a mini-batch of transitions (s, a, r, s') from $\mathcal{D}$
  // Computing advantage estimation
  Compute advantage estimation using critic network: $A_t = r_t + \gamma V(s_{t+1}) - V(s_t)$
  // Updating the policy network
  Update actor network by optimizing PPO loss: $\mathcal{L}_{\text{PPO}}(\theta)$
  // Updating the value function network
  Update critic network by minimizing squared error: $(V(s_t) - y_t)^2$
  // Updating the replay buffer
  Update experience replay buffer $\mathcal{D}$ with new transitions
end for
Output: Trained actor network with parameters θ

Hyperparameter	**Value**
Number of LSTM layers	3
LSTM units	256
Hidden units	128
PPO clipping range	0.2
Learning rate	3×10^{-4}
Replay buffer size	50000
Mini-batch size	64
ϵ (Exploration factor)	0.1
Max n-gram length	4
Training iterations	10000

Table 1: Hyperparameters

idea behind the generation of this dataset is to extract English sentences from scholarly papers according to specific rules, and then translate them into Japanese using a high-quality machine translation model. Subsequently, these sentences are transcribed back into English by non-native English speakers from Japan.

In terms of evaluation metrics, in addition to the GLUE, we sought to provide a more comprehensive assessment of our model. We, therefore, calculated the F0.5 scores using ERRANT (Bryant et al., 2017) and the Perplexity (PPL) was also calculated utilizing the Natural Language Toolkit (NLTK)[2] for the purpose of evaluating the model.

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

Model	GLUE	P	R	F0.5	PPL
H-ND	9.5	5.4	2.9	4.6	406
ED-ND	23.8	21.8	12.8	19.2	236
GEC	7.3	22.2	6.2	14.6	414
NRL (Our)	**35.85**	**29.2**	**14.1**	**24.0**	**225**

Table 2: Results of quantitative evaluation

Results The experimental results are shown in Table 2, An example output comparison is shown in Table 3.

Analysis The evaluation demonstrates that our NRL model has exhibited improvements across all performance metrics in comparison to three baseline models, thereby validating the efficacy of our method. In Example 3, as illustrated in Table 3, our model has transcribed ″in all documents″ in the Draft as ″for the whole document.″ Though it deviates by a single word from the reference ″for a whole document,″ this deviation nonetheless underscores a more potent transcription capability in our model compared to the baseline models. Furthermore, the determination of whether to use ″a″ or ″the″ in this instance cannot be ascertained solely from this sentence, as it requires contextual comprehension, which is beyond the objective of the SentRev task. Therefore, although there remains a discrepancy with the reference, we consider the current output to be quite ideal, given the characteristics of the SentRev task.

Our initial conjecture was that the continu-

Draft	The global modeling using the reinforcement learning in all documents is our work in the future.
H-ND	The global modeling of the reinforcement learning using all documents in our work is the future.
ED-ND	In our future work, we plan to explore the use of global modeling for reinforcement learning in all documents.
GEC	Global modelling using reinforcement learning in all documents is our work in the future.
NRL (Our)	The global modelling using reinforcement learning for the whole document is a future work.
Reference	The global modeling using reinforcement learning for a whole document is our future work.

Table 3: A example of Comparison of Different Model Outputs

ous learning and self-enhancement attributes of NRL would align with the incremental rewriting characteristics of SentRev. Consequently, we hypothesized that the NRL model might perform critical transcription on some key parts of the Draft, a supposition that has now been corroborated. On the other hand, we opted for GLUE as our reward function, and the evaluation has substantiated that this can effectively enhance the fluency of English sentences.

5 Conclusion and future work

In this study, we introduce a meticulously crafted method of NRL for the application in SentRev. Our approach outperforms three baseline methods across multiple metrics, illustrating a more congruent alignment of reinforcement learning techniques with SentRev. Simultaneously, this research exposes the limitations of SentRev in acquiring sentence-level knowledge, which constrains its ability to capture the contextual nuances within paragraphs of a text, thus manifesting pronounced limitations in enhancing the fluency of English text. Despite these considerable challenges, we contemplate an attempt at paragraph-level rewriting in future works, enhancing the fluency of English writing at a higher dimensional level.

6 Acknowledgements

This study was conducted with the support of Future Robotics Organization, Waseda University, and as a part of the humanoid project at the Humanoid Robotics Institute, Waseda University. This work was supported by JSPS KAKENHI Grant Numbers JP20H04267, JP21H05055. This work was also supported by a Waseda University Grant for Special Research Projects (Project number: 2023C-179).

References

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Hongyu Gong, Suma Bhat, Lingfei Wu, Jinjun Xiong, and Wen mei Hwu. 2019. Reinforcement learning based text style transfer without parallel training corpus.

Shixiang Gu, Timothy Lillicrap, Ilya Sutskever, and Sergey Levine. 2016. Continuous deep q-learning with model-based acceleration. In *International conference on machine learning*, pages 2829–2838. PMLR.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation*, 9(8):1735–1780.

Takumi Ito, Tatsuki Kuribayashi, Hayato Kobayashi, Ana Brassard, Masato Hagiwara, Jun Suzuki, and Kentaro Inui. 2019. Diamonds in the rough: Generating fluent sentences from early-stage drafts for academic writing assistance. In *Proceedings of the 12th International Conference on Natural Language Generation*, pages 40–53, Tokyo, Japan. Association for Computational Linguistics.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space.

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):529–533.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

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. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 48–53, Minneapolis, Minnesota. Association for Computational Linguistics.

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372, Taipei, Taiwan. Asian Federation of Natural Language Processing.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Felix Stahlberg and Shankar Kumar. 2020. Seq2Edits: Sequence transduction using span-level edit operations. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5147–5159, Online. 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. *Advances in neural information processing systems*, 30.

Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel Bowman. 2018. GLUE: A multi-task benchmark and analysis platform for natural language understanding. In *Proceedings of the 2018 EMNLP Workshop BlackboxNLP: Analyzing and Interpreting Neural Networks for NLP*, pages 353–355, Brussels, Belgium. Association for Computational Linguistics.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3612–3621, Brussels, Belgium. Association for Computational Linguistics.

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, San Diego, California. Association for Computational Linguistics.

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. 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 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.