Bidirectional Transformer Reranker for Grammatical Error Correction

Ying Zhang[1], **Hidetaka Kamigaito**[2], **and Manabu Okumura**[1,3]
[1]Tokyo Institute of Technology
[2]NARA Institute of Science and Technology
[3]RIKEN Center for Advanced Intelligence Project
{zhang,oku}@lr.pi.titech.ac.jp kamigaito.h@is.naist.jp

Abstract

Pre-trained seq2seq models have achieved state-of-the-art results in the grammatical error correction task. However, these models still suffer from a prediction bias due to their unidirectional decoding. Thus, we propose a bidirectional Transformer reranker (BTR), that re-estimates the probability of each candidate sentence generated by the pre-trained seq2seq model. The BTR preserves the seq2seq-style Transformer architecture but utilizes a BERT-style self-attention mechanism in the decoder to compute the probability of each target token by using masked language modeling to capture bidirectional representations from the target context. For guiding the reranking, the BTR adopts negative sampling in the objective function to minimize the unlikelihood. During inference, the BTR gives final results after comparing the reranked top-1 results with the original ones by an acceptance threshold λ. Experimental results show that, in reranking candidates from a pre-trained seq2seq model, T5-base, the BTR on top of T5-base could yield 65.47 and 71.27 $F_{0.5}$ scores on the CoNLL-14 and BEA test sets, respectively, and yield 59.52 GLEU score on the JFLEG corpus, with improvements of 0.36, 0.76 and 0.48 points compared with the original T5-base. Furthermore, when reranking candidates from T5-large, the BTR on top of T5-base improved the original T5-large by 0.26 points on the BEA test set.[1]

1 Introduction

Grammatical error correction (GEC) is a sequence-to-sequence task which requires a model to aim to correct an ungrammatical sentence. An example is presented in Table 1.[2] Various neural models for GEC have emerged (Junczys-Dowmunt et al., 2018; Kiyono et al., 2019; Kaneko et al., 2020; Rothe et al., 2021) due to the importance of this task for language-learners who tend to produce ungrammatical sentences.

Previous studies have shown that GEC can be approached as machine translation by using a seq2seq model (Luong et al., 2015) with a Transformer (Vaswani et al., 2017) architecture (Junczys-Dowmunt et al., 2018; Zhao et al., 2019; Kiyono et al., 2019; Kaneko et al., 2020; Rothe et al., 2021). As a neural model consists of an encoder and a decoder, the seq2seq architecture typically requires a large amount of training data. Because GEC suffers from limited training data, applying a seq2seq model for GEC results in a low-resource setting, that can be handled by introducing synthetic data for training (Kiyono et al., 2019; Omelianchuk et al., 2020; Stahlberg and Kumar, 2021). However, as pointed out by Rothe et al. (2021), the use of synthetic data in GEC may result in a distributional shift and require language-specific tuning, which can be time-consuming and resource-intensive.

Considering the limitations of the synthetic data, the current trend is to utilize the learned and general representations from a pre-trained model, such as BERT (Devlin et al., 2019), XLNet (Yang et al., 2019), BART (Lewis et al., 2020), and T5 (Raffel et al., 2020), which have been trained on large corpora and shown to be effective for various downstream tasks. According to Kaneko et al. (2020), incorporating a pre-trained masked language model (MLM) into a seq2seq model could facilitate correction. In addition, as reported by Rothe et al. (2021), the pre-trained T5 model achieved state-of-the-art results on GEC benchmarks for four languages after successive fine-tuning with the cleaned LANG-8 corpus (cLang-8) (Rothe et al., 2021).

Although the seq2seq model with pre-trained representations has shown to be effective for GEC, its performance was still constrained by its unidirectional decoding. As suggested by Liu et al. (2021), for an ungrammatical sentence, a fully pre-

[1]Our code is available at https://github.com/zhangying9128/BTR.

[2]Appendix L presents more examples from the CoNLL-14 (Ng et al., 2014) and JFLEG test sets.

Findings of the Association for Computational Linguistics: ACL 2023, pages 3801–3825
July 9-14, 2023

Source	Speed camera can be placed in many locations along a highway .
Gold 1	Speed cameras can be placed in many locations along a highway .
Candidate 1 (RoBERTa, T5GEC)	A speed camera can be placed in many locations along a highway .
Candidate 2 (BTR, R2L)	Speed cameras can be placed in many locations along a highway .
Candidate 3	A Speed camera can be placed in many locations along a highway .

Table 1: Examples of reranked outputs from the JFLEG (Napoles et al., 2017) test set. The 3 candidate sentences were generated by T5GEC (§5.1). Blue indicates the range of corrections. "Candidate 1 (T5GEC)" denotes that T5GEC regards "Candidate 1" as the most grammatical correction.

trained seq2seq GEC model (Kiyono et al., 2019) could generate several high-quality grammatical sentences by beam search. However, even among these candidates, there may be still a gap between the selected hypothesis and the most grammatical one. Our experimental results, listed in Table 5, also demonstrate their investigation. To solve this decoding problem, given the hypotheses of a seq2seq GEC model, Kaneko et al. (2019) used BERT to classify between ungrammatical and grammatical hypotheses, and reranked them on the basis of the classification results. The previous studies (Kiyono et al., 2019; Kaneko et al., 2020) also showed that the seq2seq GEC model decoding in an opposite direction, i.e., right-to-left, is effective as a reranker for a left-to-right GEC model.

Therefore, to further improve the performance of the pre-trained seq2seq model for GEC, it is essential to find ways to leverage the bidirectional representations of the target context. In this study, on the basis of the seq2seq-style Transformer model, we propose a bidirectional Transformer reranker (BTR) to handle the interaction between the source sentence and the bidirectional target context. The BTR utilizes a BERT-style self-attention mechanism in the decoder to predict each target token by masked language modeling (Devlin et al., 2019). Given several candidate target sentences from a base model, the BTR can re-estimate the sentence probability for each candidate from the bidirectional representation of the candidate, which is different from the conventional seq2seq model. During training, for guiding the reranking, we adopt negative sampling (Mikolov et al., 2013) for the objective function to minimize the unlikelihood while maximizing the likelihood. In inference, considering the robustness of pre-trained models, we compare the reranked top-1 results with the original ones by an acceptance threshold λ to decide whether to accept the suggestion from the BTR.

We regard the state-of-the-art model for GEC (Rothe et al., 2021), a pre-trained Transformer model, T5 (either T5-base or T5-large), as our base model and utilize its generated candidates for reranking. Because the BTR can inherit learned representations from a pre-trained Transformer model, we construct the BTR on top of T5-base. Our experimental results showed that, by reranking candidates from a fully pre-trained and fine-tuned T5-base model, the BTR on top of T5-base can achieve an $F_{0.5}$ score of 65.47 on the CoNLL-14 benchmark. The BTR on top of T5-base also outperformed T5-base on the BEA test set by 0.76 points, achieving an $F_{0.5}$ score of 71.27. Adopting negative sampling for the BTR also generated a peaked probability distribution for ranking, and so grammatical suggestions could be selected by using λ. Furthermore, the BTR on top of T5-base was robust even when reranking candidates from T5-large and improved the performance by 0.26 points on the BEA test set.

2 Related Work

For directly predicting the target corrections from corresponding input tokens, Omelianchuk et al. (2020) and Malmi et al. (2022) regarded the encoder of the Transformer model as a non-autoregressive GEC sequence tagger. The experimental results of Omelianchuk et al. (2020) showed that, compared with the randomly initialized LSTM (Hochreiter and Schmidhuber, 1997), the pre-trained models, such as RoBERTa (Liu et al., 2019), GPT-2 (Radford et al., 2019), and ALBERT (Lan et al., 2020), can achieve higher $F_{0.5}$ scores as a tagger. Sun et al. (2021) considered GEC as a seq2seq task and introduced the Shallow Aggressive Decoding (SAD) for the decoder of the Transformer. With the SAD, the performance of a pre-trained seq2seq model, BART, surpassed the sequence taggers of Omelianchuk et al. (2020). The T5 xxl model is a pre-trained seq2seq model with 11B parameters (Raffel et al., 2020). After fine-tuning with the cLang-8 corpus, T5 xxl and mT5 xxl (Xue et al., 2021), a multilingual version of T5, achieved state-of-the-art results on GEC bench-

marks in four languages: English, Czech, German, and Russian (Rothe et al., 2021). This demonstrated that performing a single fine-tuning step for a fully pre-trained seq2seq model is a simple and effective method for GEC without incorporating a copy mechanism (Zhao et al., 2019), the SAD or the output from a pre-trained MLM (Kaneko et al., 2020). Despite the improvements brought about by the pre-trained representations, the conventional seq2seq structure suffers from a prediction bias due to its unidirectional decoding. According to Liu et al. (2021), by using beam search, a fully pre-trained seq2seq GEC model (Kiyono et al., 2019) can generate several high-quality grammatical hypotheses, which include one that is more grammatical than the selected one.

To address the shortcoming of the unidirectional decoding, previous studies (Kiyono et al., 2019; Kaneko et al., 2019, 2020) introduced reversed representations to rerank the hypotheses. Kiyono et al. (2019) and Kaneko et al. (2020) utilized a seq2seq GEC model that decodes in the opposite direction (right-to-left) to rerank candidates, which was effective to select a more grammatical sentence than the original one. This finding motivated us to use a bidirectional decoding method for our model. Instead of using a seq2seq model, Kaneko et al. (2019) fine-tuned BERT as a reranker to evaluate the grammatical quality of a sentence. By using masked language modeling, BERT learned deep bidirectional representations to distinguish between grammatical and ungrammatical sentences. However, BERT did not account for the positions of corrections, as it discarded the source sentence and considered only the target sentence. This made it difficult for BERT, as a reranker, to recognize the most suitable corrected sentence for an ungrammatical sentence. Salazar et al. (2020) proposed the use of pseudo-log-likelihood scores (PLL) for reranking. They demonstrated that RoBERTa, with the PLL for reranking, outperformed the conventional language model GPT-2 when reranking candidates in speech recognition and machine translation tasks. Zhang et al. (2021) also claimed that the pre-trained model, MPNet (Song et al., 2020), was more effective than GPT-2 when using PLL for reranking in discourse segmentation and parsing.

Zhang and van Genabith (2021) proposed a bidirectional Transformer-based alignment (BTBA) model, which aims to assess the alignment between the source and target tokens in machine translation. To achieve this, BTBA masked and predicted the current token with attention to both left and right sides of the target context to produce alignments for the current token. Specifically, to assess alignments from the attention scores in all cross-attention layers, the decoder in BTBA discarded the last feed-forward layer of the Transformer model and directly predicted masked tokens from the output of the last cross-attention layer. Even though the target context on both sides was taken into consideration, one limitation of BTBA was that the computed alignments ignored the representation of the current token. To produce more accurate alignments, Zhang and van Genabith (2021) introduced full context based optimization (FCBO) for fine-tuning, in which BTBA no longer masks the target sentence to use the full target context.

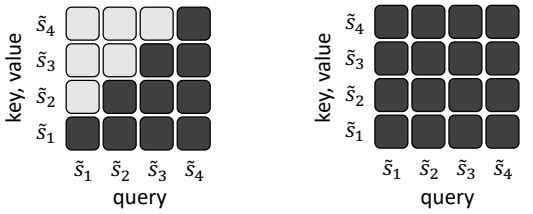
In our research, to determine the most appropriate correction for a given erroneous sentence, we model the BTR as a seq2seq reranker, which encodes the erroneous sentence using an encoder and decodes a corrected sentence using a decoder. In contrast to the conventional seq2seq model, we use masked language modeling to mask and predict each target token in the decoder and estimate the sentence probability for each candidate using PLL. Unlike BTBA, the BTR preserves the last feed-forward layer in the decoder to predict masked tokens more accurately. Because the original data of the masked tokens should be invisible in the prediction, the FCBO fine-tuning step is not used in the BTR. Compared with BTBA, the BTR keeps the structure of the Transformer model and can easily inherit parameters from pre-trained models.

3 Preliminary

Because the decoder of the BTR uses masked language modeling to rerank candidates based on the PLL, in this section, we explain how a Transformer-based GEC model generates the candidates, the masked language modeling used in BERT, and how to compute the PLL.

3.1 Transformer-based GEC Model

Given an ungrammatical sentence $\boldsymbol{x} = (x_1, \ldots, x_n)$, a GEC model corrects $\boldsymbol{x}$ into its grammatical sentence $\boldsymbol{y} = (y_1, \ldots, y_m)$, where x_i is the i-th token in $\boldsymbol{x}$ and y_j is the j-th token in $\boldsymbol{y}$. As an auto-regressive model, a Transformer-based GEC model with parameter θ



(a) Causal (b) Fully-visible

Figure 1: Mask patterns in the Transformer model (Vaswani et al., 2017) (a) and in the BTR (b) for the self-attention mechanism in the decoder. Light cells indicate no attention.

Method	Gold (%)	Unique (%)	Oracle ($F_{0.5}$)
Nucleus sampling	28.70	43.61	49.27
Top-k sampling	29.62	48.57	48.40
Beam search	**37.97**	**98.93**	**55.11**
Diverse beam search	28.46	38.78	50.39

Table 2: Results for the T5GEC on the CoNLL-13 corpus with various decoding methods.

decomposes $p(\boldsymbol{y}|\boldsymbol{x};\theta)$ as follows:

$$\log p(\boldsymbol{y}|\boldsymbol{x};\theta) = \sum_{j=1}^{m} \log p(y_j|\boldsymbol{x}, \boldsymbol{y}_{<j};\theta), \quad (1)$$

$$p(y_j|\boldsymbol{x}, \boldsymbol{y}_{<j};\theta) = \mathrm{softmax}(W\tilde{s}_j + b), \quad (2)$$

where $\tilde{s}_j$ is the final hidden state from the decoder at the j-th decoding step. W is a weight matrix, b is a bias term, and $\boldsymbol{y}_{<j}$ denotes $(y_1, \ldots, y_{j-1})$. $\tilde{s}_j$ is computed as described in Appendix A.

3.2 Decoding Method

The pre-trained T5 model with Transformer architecture achieved state-of-the-art results in GEC by using beam search for decoding (Rothe et al., 2021). However, previous studies (Li and Jurafsky, 2016; Vijayakumar et al., 2018) have suggested that beam search tends to generate sequences with slight differences. This can constrain the upper bound score when reranking candidates (Ippolito et al., 2019). To select the optimal decoding method for a Transformer-based GEC Model, T5GEC, we compared beam search with diverse beam search (Vijayakumar et al., 2018), top-k sampling (Fan et al., 2018), and nucleus sampling (Holtzman et al., 2020). For each pair of data in CoNLL-13 corpus (Ng et al., 2013), we required all decoding methods to generate 5 candidate sequences with a maximum sequence length of 200. When using diverse beam search, we fixed the beam group and diverse penalty to 5 and 0.4, respectively. Meanwhile, we set the top-k as 50 and the top-p as 0.95 for top-k sampling and nucleus sampling, respectively.

Table 2 presents the compared results among different decoding methods. Oracle indicates the upper bound score that can be achieved with the generated candidates. If the candidates include the correct answer, we assume the prediction is correct. Unique (%) indicates the rate of unique sequences among all candidates. Gold (%) indicates the rate of pairs of data whose candidates include the correct answer. The results show that beam search generates more diverse sentences with the highest Oracle score compared to nucleus sampling, top-k sampling, and diverse beam search. This may be because, in the GEC task, most of the tokens in the target are the same as the source, which causes a peaked probabilities distribution to focus on one or a small number of tokens. And thus, a top-k filtering method like beam search generates more diverse sentences than sampling or using probability as a diverse penalty. Based on these results, we have chosen beam search as the decoding method for T5GEC during inference. For evaluating T5GEC, it generates the top-ranked hypothesis with a beam size of 5. To generate the top-a candidates $\mathcal{Y}_a = \{\boldsymbol{y}_1, \ldots, \boldsymbol{y}_a\}$ for reranking, it generates hypotheses with a beam size of a and a maximum sequence length of 128 and 200 for the datasets in training and prediction, respectively.

3.3 Masked Language Modeling

Masked language modeling, used in BERT, was introduced to learn bidirectional representations for a given sentence $\boldsymbol{x}$ through self-supervised learning (Devlin et al., 2019). Before pre-training, several tokens in $\boldsymbol{x}$ are randomly replaced with the mask token <M>. Let κ denote the set of masked positions, $\boldsymbol{x}_\kappa$ the set of masked tokens, and $\boldsymbol{x}_{\backslash\kappa}$ the sentence after masking. The model parameter θ is optimized by maximizing the following objective:

$$\log p(\boldsymbol{x}_\kappa|\boldsymbol{x}_{\backslash\kappa};\theta) \approx \sum_{k\in\kappa} \log p(x_k|\boldsymbol{x}_{\backslash\kappa};\theta). \quad (3)$$

Similar to Eq. (2), a linear transformation with a softmax function is utilized for the final hidden state $\tilde{h}_k$ to predict $p(x_k|\boldsymbol{x}_{\backslash\kappa};\theta)$. $\tilde{h}_k$ is computed as described in Appendix B. The corresponding PLL for $\boldsymbol{x}$ is computed by

$$\mathrm{PLL}(\boldsymbol{x};\theta) := \sum_{i=1,\kappa=\{i\}}^{|\boldsymbol{x}|} \log p(x_i|\boldsymbol{x}_{\backslash\kappa};\theta), \quad (4)$$

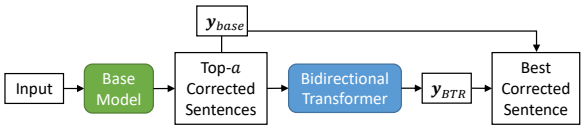


Figure 2: Overview of the reranking procedure by using the bidirectional Transformer reranker (BTR).

where $|\boldsymbol{x}|$ is the length of $\boldsymbol{x}$. As suggested by Salazar et al. (2020), when using PLL to estimate the cross-entropy loss, the loss of $x_i|\boldsymbol{x}_{\setminus\kappa}$ versus i from BERT is flatter than GPT-2, that uses the chain rule. Considering the candidate sentences might have different lengths, PLL is ideal for reranking.

4 Bidirectional Transformer Reranker (BTR)

The BTR uses masked language modeling in the decoder to estimate the probability of a corrected sentence. Given an ungrammatical sentence $\boldsymbol{x}$, a base GEC model first generates the top-a corrected sentences $\mathcal{Y}_a$, as described in Section 3.1. Assume $\boldsymbol{y}_{base} \in \mathcal{Y}_a$ is the top-ranked hypothesis from the base GEC model. The BTR selects and accepts the most optimal corrected sentence $\boldsymbol{y}_{BTR}$ from $\mathcal{Y}_a$ on the basis of the estimated sentence probability, as described in the following. Figure 2 shows the overview of the BTR for the whole procedure.

4.1 Target Sentence Probability

As PLL has been effective in estimating the sequence probability for reranking, we decompose the conditional sentence probability of $\boldsymbol{y}$ as:

$$\begin{aligned}&\log p(\boldsymbol{y}|\boldsymbol{x};\theta)\\ \approx&\mathrm{PLL}(\boldsymbol{y}|\boldsymbol{x};\theta) = \sum_{j=1,\kappa=\{j\}}^{|\boldsymbol{y}|} \log p(y_j|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta). \end{aligned} \quad (5)$$

As in Eq. (2), a linear transformation with the softmax function is utilized for the final hidden state $\tilde{s}_j$ to predict $p(y_j|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta)$.

Same as the Transformer architecture, $\tilde{s}_j$ is the result of s_j after the cross-attention and feed-forward layers. We assume the decoder consists of L layers. To capture the bidirectional representation, for $\ell \in L$, we compute s_j^ℓ as:

$$s_j^\ell = \mathrm{Attn}_s(\tilde{s}_j^{\ell-1}, \widetilde{S}_{\setminus\kappa}^{\ell-1}, \widetilde{S}_{\setminus\kappa}^{\ell-1}), \quad (6)$$

where $\tilde{s}_j^0$ is the embedding of the $j-1$-th word in $\boldsymbol{y}_{\setminus\kappa}$ and $\tilde{s}_1$ is the state of the start token <s>. $\widetilde{S}_{\setminus\kappa}^{\ell-1} = (\tilde{s}_1^{\ell-1}, \ldots, \tilde{s}_{m+1}^{\ell-1})$ denotes a set of hidden states for the joint sequence of <s> and $\boldsymbol{y}_{\setminus\kappa}$. Attn_s indicates the self-attention layer. Figure 1b shows our fully-visible attention mask for computing S^ℓ in parallel. The procedure of using the BTR to predict $p(y_j|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta)$ is shown in Appendix C.

4.2 Objective Function

As a reranker, for a given ungrammatical sentence $\boldsymbol{x}$, the BTR should compare all corresponding corrected sentences $\mathcal{Y}$ and select the most grammatical one. However, considering all possible corrected sentences for $\boldsymbol{x}$ is intractable, as suggested by Stahlberg and Byrne (2019), so we consider a subset of sequences $\mathcal{Y}_a$ based on the top-a results from the base GEC model instead.

Let $\boldsymbol{y}_{gold} \in \mathcal{Y}$ denote the gold correction for $\boldsymbol{x}$. For $\boldsymbol{y} \in \mathcal{Y}_a \cup \{\boldsymbol{y}_{gold}\}$, we follow the setting of BERT to randomly mask 15% of $\boldsymbol{y}$ and denote κ as the set of masked positions. As a result, the distribution of the masked tokens satisfies the 8:1:1 masking strategy. Following previous research (Welleck et al., 2019; Zhang et al., 2021; Song et al., 2021), given the masked sentence $\boldsymbol{y}_{\setminus\kappa}$, the model parameter θ of the BTR is optimized by maximizing the likelihood and minimizing the unlikelihood as:

$$\begin{aligned}&\log p(\boldsymbol{y}_\kappa|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta)\\ \approx&\sum_{k\in\kappa}[\mathbb{1}_{\boldsymbol{y}} \log p(y_k|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta)\\ &+ (1-\mathbb{1}_{\boldsymbol{y}}) \log(1-p(y_k|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta))],\end{aligned} \quad (7)$$

where $p(y_k|\boldsymbol{x},\boldsymbol{y}_{\setminus\kappa};\theta)$ is computed as in Section 4.1. $\mathbb{1}_{\boldsymbol{y}}$ is an indicator function defined as follows:

$$\mathbb{1}_{\boldsymbol{y}} := \begin{cases} 1 & \text{if } \boldsymbol{y} = \boldsymbol{y}_{gold} \\ 0 & \text{if } \boldsymbol{y} \neq \boldsymbol{y}_{gold} \end{cases}. \quad (8)$$

4.3 Inference

In inference, for $\boldsymbol{y} \in \mathcal{Y}_a$, the BTR scores $\boldsymbol{y}$ by

$$f(\boldsymbol{y}|\boldsymbol{x}) = \frac{\exp(\mathrm{PLL}(\boldsymbol{y}|\boldsymbol{x};\theta)/|\boldsymbol{y}|)}{\sum_{\boldsymbol{y}'\in\mathcal{Y}_a}\exp(\mathrm{PLL}(\boldsymbol{y}'|\boldsymbol{x};\theta)/|\boldsymbol{y}'|)}. \quad (9)$$

Hereafter, we denote $\boldsymbol{y}_{BTR} \in \mathcal{Y}_a$ as the candidate with the highest score $f(\boldsymbol{y}_{BTR}|\boldsymbol{x})$ for given $\boldsymbol{x}$ in the BTR. Here, $f(\boldsymbol{y}|\boldsymbol{x})$ is also considered to indicate the confidence of the BTR. Because the BTR is optimized with Eq. (7), a high score for $\boldsymbol{y}_{BTR}$ indicates a confident prediction while a low score indicates an unconfident prediction.

Considering that we build the base GEC model from a fully pre-trained seq2seq model and the

BTR from an insufficiently pre-trained model, we introduce an acceptance threshold λ to decide whether to accept the suggestion from the BTR. We accept $\boldsymbol{y}_{BTR}$ only when it satisfies the following equation; otherwise, $\boldsymbol{y}_{base}$ is still the final result:

$$f(\boldsymbol{y}_{BTR}|\boldsymbol{x}) - f(\boldsymbol{y}_{base}|\boldsymbol{x}) > \lambda, \quad (10)$$

where λ is a hyperparameter tuned on the validation data.

5 Experiments

5.1 Compared Methods

We evaluated the BTR as a reranker for two versions of candidates, normal and high-quality ones, generated by two seq2seq GEC models, T5GEC and T5GEC (large). We compared the BTR with three other rerankers, R2L, BERT, and RoBERTa.

T5GEC: We used the state-of-the-art model (Rothe et al., 2021) as our base model for GEC. This base model inherited the pre-trained T5 version 1.1 model (T5-base) (Raffel et al., 2020) and was fine-tuned as described in Section 3.1. We denote this base model as T5GEC hereafter. Although the T5 xxl model yielded the most grammatical sentences in Rothe et al. (2021), it contained 11B parameters and was not suitable for our current experimental environment. Thus, we modeled T5GEC on top of a 248M-parameter T5-base model. To reproduce the experimental results of Rothe et al. (2021), we followed their setting and fine-tuned T5GEC once with the cLang-8 dataset.

T5GEC (large): To investigate the potential of the BTR for reranking high-quality candidates, we also fine-tuned one larger T5GEC model with a 738M-parameter T5-large structure. We denote this model as T5GEC (large).

R2L: The decoder of the conventional seq2seq model can generate a target sentence either in a left-to-right or right-to-left direction. Because T5GEC utilized the left-to-right direction, and previous research (Sennrich et al., 2016; Kiyono et al., 2019; Kaneko et al., 2020) showed the effectiveness of reranking using the right-to-left model, we followed Kaneko et al. (2020) to construct four right-to-left T5GEC models, which we denote as R2L. R2L reranks candidates based on the sum score of the base model (L2R) and ensembled R2L.

BERT: We followed Kaneko et al. (2019) to fine-tune four BERT with 334M parameters. During fine-tuning, both source and target sentences were annotated with either <0> (ungrammatical) or <1> (grammatical) label for BERT to classify. During inference, the ensembled BERT reranks candidates based on the predicted score for the <1> label.

RoBERTa: We fine-tuned four 125M parameters RoBERTa to compare our bidirectional Transformer structure with the encoder-only one. During fine-tuning, the source and target sentences were concatenated, and RoBERTa masked and predicted only the target sentence as the BTR. During prediction, the ensembled RoBERTa reranks candidates with the acceptance threshold λ as the BTR.

5.2 Setup for the BTR

Because there was no pre-trained seq2seq model with a self-attention mechanism for masked language modeling in the decoder, we constructed the BTR using the 248M T5 model (T5-base) and pre-trained it with the Realnewslike corpus (Zellers et al., 2019). To compare the BTR with R2L, we also constructed R2L using T5-base, and pre-trained both models as follows. To speed up pre-training, we initialized the BTR and R2L model parameters with the fine-tuned parameters θ of T5GEC. During pre-training, we followed Raffel et al. (2020) for self-supervised learning with a span masking strategy. Specifically, 15% of the tokens in a given sentence were randomly sampled and removed. The input sequence was constructed by the rest tokens while the target sequence was the concatenation of dropped-out tokens. An example is provided in Table 3. We pre-trained the BTR and R2L with $65536 = 2^{16}$ and 10000 steps, respectively. Because the BTR masked and predicted only 15% of the tokens in Eq. (7), the true steps for the BTR were $2^{16} \times 0.15 \approx 10000$. We used a maximum sequence length of 512 and a batch size of $2^{20} = 1048576$ tokens. In total, we pre-trained $10000 \times 2^{20} \approx 10.5$B tokens, which were less than the pre-trained T5 with 34B tokens. The pre-training for R2L and the BTR took 2 and 13 days, respectively, with 2 NVIDIA A100 80GB GPUs. This indicates the BTR requires more training time and resources than R2L. We provide a plot of the pre-training loss in Appendix D.

After pre-training, we successively fine-tuned the BTR with the cLang-8 dataset. Like R2L, BERT, and RoBERTa, our fine-tuned BTR is the ensemble of four models with random seeds.

5.3 Datasets

For fair comparison, we pre-trained R2L and the BTR with the Realnewslike corpus. This corpus

Model	Inputs	Targets
	Self-supervised learning for pre-training	
BERT / RoBERTa	Thank you so <M> me to your party <M> week .	Thank you for inviting me to your party last week .
T5 / R2L	Thank you <X> me to your party <Y> week .	<X> for inviting <Y> last <Z>
BTR	Thank you <X> me to your party <Y> week .	<X> for <M> you last <Z>
	Supervised learning for fine-tuning	
BERT	Thank you for inviting me to your party last week .	<1>
T5 / R2L	Thank you for invite me to your party last week .	Thank you for inviting me to your party last week .
BTR / RoBERTa	Thank you for invite me to your party last week .	Thank you so <M> me to your party <M> week .

Table 3: Examples of data pairs for self-supervised and supervised learning used by each model. The grammatical text is "Thank you for inviting me to your party last week ." <M> denotes a mask token. <X>, <Y>, and <Z> denote sentinel tokens that are assigned unique token IDs. <1> denotes the input sentence is classified as a grammatical sentence. Red indicates an error in the source sentence while Blue indicates a token randomly replaced by the BERT-style masking strategy.

Dataset	Usage	Lang	# of data (pairs)
Realnewslike	pre-train	EN	148,566,392
cLang-8	train	EN	2,372,119
CoNLL-13 (cleaned)	valid	EN	1,381
CoNLL-14	test	EN	1,312
BEA	test	EN	4,477
JFLEG	test	EN	747

Table 4: Dataset sizes.

contains 37 GB of text data and is a subset of the C4 corpus (Raffel et al., 2020). To shorten the input and target sequences, we split each text into paragraphs. During fine-tuning, we followed the steps of Rothe et al. (2021) and regarded the cLang-8 corpus as the training dataset.

While the CoNLL-13 dataset was used for validation, the standard benchmarks from JFLEG, CoNLL-14, and the BEA test set (Bryant et al., 2019) were used for evaluation. While the CoNLL-14 corpus considers the minimal edit of corrections, JFLEG evaluates the fluency of a sentence. The BEA corpus contains much more diverse English language levels and domains than the CoNLL-14 corpus. We used a cleaned version of CoNLL-13 with consistent punctuation tokenization styles. Appendix E lists our cleaning steps and the experimental results on the cleaned CoNLL-14 set. Table 4 summarizes the data statistics.

5.4 Evaluation Metrics

The evaluation on the BEA test set was automatically executed in the official BEA-19 competition in terms of span-based correction $F_{0.5}$ using the ERRANT (Bryant et al., 2017) scorer. For the CoNLL-13 and 14 benchmarks, we evaluated the correction $F_{0.5}$ using the official M^2 (Dahlmeier and Ng, 2012) scorer. For the JFLEG corpus, we evaluated the GLEU (Napoles et al., 2015).

We report only significant results on the CoNLL-14 set, because the gold data for the BEA test set is unavailable, and the evaluation metric GLEU for the JFLEG test set requires a sampling strategy for multiple references. We used the paired t-test to evaluate whether the difference between $\boldsymbol{y}_{BTR}$ and $\boldsymbol{y}_{base}$ on the CoNLL-14 set is significant, as only limited $\boldsymbol{y}_{BTR}$ differed from $\boldsymbol{y}_{base}$ among the suggestions from the BTR.

5.5 Hyperparameters

Appendix F lists our hyperparameter settings for pre-training and fine-tuning each model.

We followed the setting of Zhang et al. (2021) to separately tune a for training and prediction, based on the model performance on the validation dataset with candidates generated by T5GEC. We denote a for training and prediction as a_{train} and a_{pred}, respectively. The threshold (λ) for the BTR and RoBERTa was tuned together with a. We set a_{train} to 20, 0 for the BTR and RoBERTa, respectively, and a_{pred} was set to 5 for all rerankers. When $a_{train} = 20$, λ was set to 0.4 and 0.8 with respect to the candidates generated by T5GEC and T5GEC (large), respectively. When $a_{train} = 0$, λ for the RoBERTa was set to 0.1 for the two versions of candidates. The experimental results for tuning a_{train}, a_{pred}, and λ are listed in Appendix G.

5.6 Results

Table 5 presents our main results.[3] While reranking by R2L yielded the highest $F_{0.5}$ score of 71.42 on the BEA test set, it yielded only a lower score

[3]The mean and standard deviation results of the BTR, R2L, and RoBERTa are listed in Appendix H.

Model	CoNLL-13			CoNLL-14			BEA			JFLEG
	p	r	$F_{0.5}$	p	r	$F_{0.5}$	p	r	$F_{0.5}$	GLEU
Oracle	65.50	33.71	55.11	73.74	51.38	67.87	-	-	-	61.13
T5GEC *	-	-	-	-	-	65.13	-	-	69.38	-
T5GEC	59.19	29.65	49.36	71.27	48.37	65.11	73.96	59.45	70.51	59.04
R2L	**60.94**	29.14	50.02	**71.87**	46.81	64.92	**75.51**	58.69	**71.42**	58.93
w/o L2R	59.56	28.97	49.19	71.36	46.68	64.54	73.51	57.96	69.76	58.69
BERT	44.53	**35.74**	42.44	55.93	**53.18**	55.36	49.91	**64.37**	52.26	55.69
RoBERTa ($\lambda = 0.1$) w/o a_{train}	59.20	29.63	49.35	71.14	48.42	65.04	74.04	59.37	70.55	59.17
w/o a_{train}, λ	54.83	28.88	46.48	65.64	47.24	60.90	65.85	57.71	64.05	57.49
BTR ($\lambda = 0.4$)	59.87	30.54	**50.22**	71.62	48.74	**65.47**	74.68	60.27	71.27	59.17
w/o λ	58.10	30.37	49.13	69.52	48.07	63.82	72.69	60.71	69.93	**59.52**
w/o a_{train}, λ	51.30	30.94	45.34	62.83	49.03	59.48	64.35	60.74	63.60	57.62

Table 5: Results for the models on each dataset with candidates from T5GEC. * indicates the score presented in Rothe et al. (2021). Bold scores represent the highest (p)recision, (r)ecall, $F_{0.5}$, and GLEU for each dataset.

Candidate	Accept	Reject	Equal
Proportion(%)	12.50	21.11	66.39
$\boldsymbol{y}_{base}$	61.67	**61.66**†	**68.78**
$\boldsymbol{y}_{BTR}$	**63.97**†	57.28	**68.78**

Table 6: Results for the BTR ($\lambda = 0.4$) on CoNLL-14 with candidates from T5GEC. $\boldsymbol{y}_{base}$ and $\boldsymbol{y}_{BTR}$ denote the selections by T5GEC and suggestions by the BTR, respectively. † indicates that the difference between $\boldsymbol{y}_{BTR}$ and $\boldsymbol{y}_{base}$ is significant with a p-value < 0.05. Bold scores represent the highest $F_{0.5}$ for each case.

Model	CoNLL-13	CoNLL-14	BEA	JFLEG
Oracle	56.44	70.08	-	63.87
T5GEC (large) *	-	66.10	72.06	-
T5GEC (large)	50.79	66.83	72.15	61.88
R2L	50.87	66.68	**72.98**	61.32
RoBERTa ($\lambda = 0.1$) w/o a_{train}	50.76	**66.85**	72.20	61.85
BTR ($\lambda = 0.8$)	**51.00**	66.57	72.41	**61.97**

Table 7: Results for the models on each dataset with candidates generated by T5GEC (large). * indicates the score presented in Rothe et al. (2021). The precision and recall can be found in Appendix K.

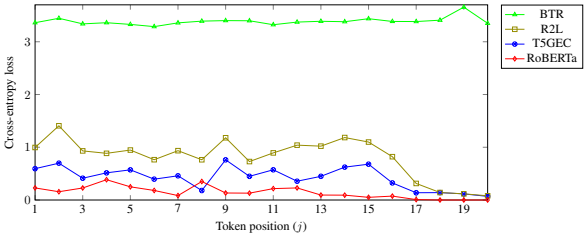


Figure 3: Cross-entropy loss of y_j versus j. The loss was averaged over CoNLL-14's 149 tokenized utterances with length in interval $[18, 20]$ (including <eos>).

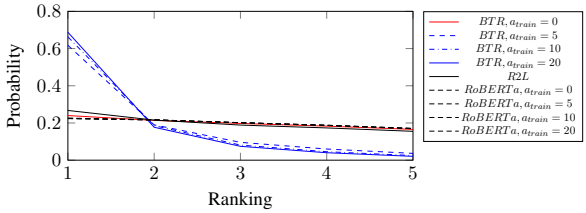


Figure 4: Average probability for each rank on the CoNLL-14 test set. The top-5 candidate sentences were generated by T5GEC.

than the BTR ($\lambda = 0.4$) on CoNLL-14 and JFLEG test sets. Meanwhile, the improvements brought by R2L depended on the beam searching score from L2R, suggesting that the unidirectional representation offers fewer gains compared to the bidirectional representation from the BTR. Reranking candidates by BERT resulted in the lower $F_{0.5}$ and GLEU scores than T5GEC. This may be because BERT considers only the target sentence and ignores the relationship between the source and the target. The BTR ($\lambda = 0.4$) achieved an $F_{0.5}$ score of 71.27 on the BEA test set.[4] On the CoNLL-14 test set, the BTR ($\lambda = 0.4$) attained the highest $F_{0.5}$ score of 65.47, with improvements of 0.36 points from T5GEC. The use of the threshold and negative candidates played an important role in the BTR. Without these two mechanisms, the BTR achieved only 59.48 and 63.60 $F_{0.5}$ scores, respectively, on the CoNLL-14 and BEA test sets, which were lower than those of the original selection. In the meantime, the BTR without the threshold could achieve the highest GLEU score of 59.52 on the JFLEG corpus, which indicates $\lambda = 0.4$ is too high for the JFLEG corpus. This is because of the different distributions and evaluation metrics between the CoNLL-13 and JFLEG corpus, as proved in Appendix J. Compared to RoBERTa ($\lambda = 0.1$) w/o a_{train} of the encoder-only structure,

[4]Experimental results in more details for different CEFR levels and error types can be found in Appendix I.

the BTR ($\lambda = 0.4$) can achieve higher $F_{0.5}$ scores on CoNLL-13, 14, and BEA test sets, and a competitive GLEU score on the JFLEG corpus. These results show the benefit of using the Transformer with the encoder-decoder architecture in the BTR.

Table 6 demonstrates the effect of using λ. *Equal* denotes the suggestion $\boldsymbol{y}_{BTR}$ is exactly $\boldsymbol{y}_{base}$. *Accept* denotes $\boldsymbol{y}_{BTR}$ satisfies Eq. (10) and $\boldsymbol{y}_{BTR}$ will be the final selection, while *Reject* denotes $\boldsymbol{y}_{BTR}$ does not satisfy the equation and $\boldsymbol{y}_{base}$ is still the final selection. Most of the final selections belonged to *Equal* and achieved the highest $F_{0.5}$ score of 68.78. This indicates the sentences in *Equal* can be corrected easily by both the BTR ($\lambda = 0.4$) and T5GEC. Around $1/3$ of the new suggestions proposed by the BTR ($\lambda = 0.4$) were accepted and achieved an $F_{0.5}$ score of 63.97, which was a 2.3-point improvement from $\boldsymbol{y}_{base}$. However, around $2/3$ of the new suggestions were not accepted, and the original selection by T5GEC resulted in a higher $F_{0.5}$ score than these rejected suggestions. These results show that, among the new suggestions, the BTR was confident only for some suggestions. The confident suggestions tended to be more grammatical, whereas the unconfident suggestions tended to be less grammatical than the original selections. Appendix J shows the analysis.

Table 7 lists the performances when reranking high-quality candidates. While R2L still achieved the highest $F_{0.5}$ score on the BEA test set, it was less effective than the BTR on the JFLEG corpus. Although the BTR ($\lambda = 0.8$) used only 248M parameters and was trained with the candidates generated by T5GEC, it could rerank candidates from T5GEC (large) and achieve 61.97 GLEU and 72.41 $F_{0.5}$ scores on the JFLEG and BEA test sets, respectively. This finding indicates the sizes of the BTR and the base model do not need to be consistent, and a smaller BTR can also work as a reranker for a larger base model. RoBERTa ($\lambda = 0.1$) w/o a_{train} achieved the highest $F_{0.5}$ score of 66.85 on the CoNLL-14 corpus with only 0.02-point improvement from T5GEC (large), which reflects the difficulty in correcting uncleaned sentences.

To investigate the difference among R2L, RoBERTa ($\lambda = 0.1$) w/o a_{train}, and the BTR ($\lambda = 0.4$), we compared the precision and recall of the three rerankers in Table 5. In most cases, R2L tended to improve the precision but lower the recall from T5GEC. The improvements brought by RoBERTa from T5GEC for both precision and recall are limited. Meanwhile, the BTR could improve both precision and recall from the original ranking. Because T5GEC already achieved a relatively high precision and low recall, there was more room to improve recall, which was demonstrated by the BTR. Figure 3 shows both T5GEC and R2L have a relatively high cross-entropy loss for tokens at the beginning positions and a low loss for tokens at the ending positions, even though the loss of R2L was the sum of two opposite decoding directions. This may be because the learning by the auto-regressive models for the latest token was over-fitting and for the global context was under-fitting, as Qi et al. (2020) indicated. RoBERTa has a flatter loss with less sharp positions than T5GEC and R2L. Meanwhile, the BTR has a flat loss, which is ideal for reranking candidate sentences with length normalization, as suggested by Salazar et al. (2020). Figure 4 shows the probability distribution of reranking. When $a_{train} > 0$, the probability distribution of the BTR becomes peaked, which indicates that using Eq. (7) to minimize the unlikelihood could increase the probability gap between the 1st-ranked candidate and the rest. Compared with the BTR, when $a_{train} > 0$, the probability distribution of RoBERTa is as flat as $a_{train} = 0$, which suggests the effectiveness of the encoder-decoder structure compared with the encoder-only one when minimizing unlikelihood.

6 Conclusion

We proposed a bidirectional Transformer reranker (BTR) to rerank several top candidates generated by a pre-trained seq2seq model for GEC. For a fully pre-trained model, T5-base, the BTR could achieve 65.47 and 71.27 $F_{0.5}$ scores on the CoNLL-14 and BEA test sets. Our experimental results showed that the BTR on top of T5-base with limited pre-training steps could improve both precision and recall for candidates from T5-base. Since using negative sampling for the BTR generates a peaked probability distribution for ranking, introducing a threshold λ benefits the acceptance of the suggestion from the BTR. Furthermore, the BTR on top of T5-base could rerank candidates generated from T5-large and yielded better performance. This finding suggests the effectiveness of the BTR even in experiments with limited GPU resources. While the BTR in our experiments lacked sufficient pre-training, it should further improve the performance with full pre-training for reranking in future.

7 Limitations

As mentioned in the previous section, up until now, there has not been a fully pre-trained seq2seq model with a BERT-style self-attention mechanism in the decoder, while the vanilla seq2seq model tends to use a left-to-right or right-to-left unidirectional self-attention. Therefore, utilizing our proposed Bidirectional Transformer Reranker (BTR) to rerank candidates from a pre-trained vanilla seq2seq model requires additional pre-training steps, which cost both time and GPU resources. Because the BTR masks and predicts only 15% of the tokens in Eq. (7), it requires more training steps than a vanilla seq2seq model. In addition, during fine-tuning, the BTR also requires additional a_{train} negative samples, which makes the fine-tuning longer. Furthermore, tuning a_{train} will be inefficient if the training is slow. In other words, training an effective BTR requires much more time than training a vanilla seq2seq model.

As a reranker, the performance of the BTR depends on the quality of candidates. There is no room for improvement by the BTR if no candidate is more grammatical than the original selection.

References

Christopher Bryant, Mariano Felice, Øistein 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, Florence, Italy. Association for Computational Linguistics.

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.

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, Montréal, Canada. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. 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 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Angela Fan, Mike Lewis, and Yann Dauphin. 2018. Hierarchical neural story generation. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia. Association for Computational Linguistics.

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

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration. In *International Conference on Learning Representations*.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Daphne Ippolito, Reno Kriz, João Sedoc, Maria Kustikova, and Chris Callison-Burch. 2019. Comparison of diverse decoding methods from conditional language models. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3752–3762, Florence, Italy. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. 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 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Masahiro Kaneko, Kengo Hotate, Satoru Katsumata, and Mamoru Komachi. 2019. TMU transformer system using BERT for re-ranking at BEA 2019 grammatical error correction on restricted track. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 207–212, Florence, Italy. Association for Computational Linguistics.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254, Online. Association for Computational Linguistics.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on*

Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Zhenzhong Lan, Mingda Chen, Sebastian Goodman, Kevin Gimpel, Piyush Sharma, and Radu Soricut. 2020. Albert: A lite bert for self-supervised learning of language representations. In *International Conference on Learning Representations*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

Jiwei Li and Dan Jurafsky. 2016. Mutual information and diverse decoding improve neural machine translation. *CoRR*, abs/1601.00372.

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

Zhenghao Liu, Xiaoyuan Yi, Maosong Sun, Liner Yang, and Tat-Seng Chua. 2021. Neural quality estimation with multiple hypotheses for grammatical error correction. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5441–5452, Online. Association for Computational Linguistics.

Thang Luong, Hieu Pham, and Christopher D. Manning. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, Lisbon, Portugal. Association for Computational Linguistics.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies: Tutorial Abstracts*, pages 1–7, Seattle, United States. Association for Computational Linguistics.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in Neural Information Processing Systems*, volume 26. Curran Associates, Inc.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Ted Briscoe, Christian Hadiwinoto, 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, Baltimore, Maryland. Association for Computational Linguistics.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 shared task on grammatical error correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

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.

Weizhen Qi, Yu Yan, Yeyun Gong, Dayiheng Liu, Nan Duan, Jiusheng Chen, Ruofei Zhang, and Ming Zhou. 2020. ProphetNet: Predicting future n-gram for sequence-to-SequencePre-training. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pages 2401–2410, Online. Association for Computational Linguistics.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi

Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Edinburgh neural machine translation systems for WMT 16. In *Proceedings of the First Conference on Machine Translation: Volume 2, Shared Task Papers*, pages 371–376, Berlin, Germany. Association for Computational Linguistics.

Haoyu Song, Yan Wang, Kaiyan Zhang, Wei-Nan Zhang, and Ting Liu. 2021. BoB: BERT over BERT for training persona-based dialogue models from limited personalized data. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 167–177, Online. Association for Computational Linguistics.

Kaitao Song, Xu Tan, Tao Qin, Jianfeng Lu, and Tie-Yan Liu. 2020. Mpnet: Masked and permuted pre-training for language understanding. In *Advances in Neural Information Processing Systems*, volume 33, pages 16857–16867. Curran Associates, Inc.

Felix Stahlberg and Bill Byrne. 2019. On NMT search errors and model errors: Cat got your tongue? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3356–3362, Hong Kong, China. Association for Computational Linguistics.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Xin Sun, Tao Ge, Furu Wei, and Houfeng Wang. 2021. Instantaneous grammatical error correction with shallow aggressive decoding. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5937–5947, Online. Association for Computational Linguistics.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Ashwin Vijayakumar, Michael Cogswell, Ramprasaath Selvaraju, Qing Sun, Stefan Lee, David Crandall, and Dhruv Batra. 2018. Diverse beam search for improved description of complex scenes. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Sean Welleck, Jason Weston, Arthur Szlam, and Kyunghyun Cho. 2019. Dialogue natural language inference. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 3731–3741, Florence, Italy. Association for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel. 2021. mT5: A massively multilingual pre-trained text-to-text transformer. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 483–498, Online. Association for Computational Linguistics.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Rowan Zellers, Ari Holtzman, Hannah Rashkin, Yonatan Bisk, Ali Farhadi, Franziska Roesner, and Yejin Choi. 2019. Defending against neural fake news. In *Advances in Neural Information Processing Systems*, volume 32. Curran Associates, Inc.

Jingyi Zhang and Josef van Genabith. 2021. A bidirectional transformer based alignment model for unsupervised word alignment. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 283–292, Online. Association for Computational Linguistics.

Ying Zhang, Hidetaka Kamigaito, and Manabu Okumura. 2021. A language model-based generative classifier for sentence-level discourse parsing. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2432–2446, Online and Punta Cana, Dominican Republic. 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.

A Computation for $\tilde{s}_j$ in Transformer

Let FNN denote a feed-forward layer and $\mathrm{Attn}(q, K, V)$ the attention layer, where q, K, and V indicate the query, key, and value, respectively. We assume the decoder consists of L layers. To compute $\tilde{s}_j$, the encoder first encodes $\boldsymbol{x}$ into its representation $\widetilde{H}$. Then, for $\ell \in L$, the hidden state $\tilde{s}_j^\ell$ of the ℓ-th layer in the decoder is computed by

$$s_j^\ell = \mathrm{Attn}_s(\tilde{s}_j^{\ell-1}, \widetilde{S}_{\leq j}^{\ell-1}, \widetilde{S}_{\leq j}^{\ell-1}), \quad (11)$$

$$\hat{s}_j^\ell = \mathrm{Attn}_c(s_j^\ell, \widetilde{H}, \widetilde{H}), \quad (12)$$

$$\tilde{s}_j^\ell = \mathrm{FNN}(\hat{s}_j^\ell), \quad (13)$$

where $\tilde{s}_j^0$ is the embedding of the token y_{j-1} and $\tilde{s}_1$ is the state for the special token <s>, that indicates the start of a sequence. $\widetilde{S}_{\leq j}^{\ell-1}$ denotes a set of hidden states $(\tilde{s}_1^{\ell-1}, \ldots, \tilde{s}_j^{\ell-1})$. Attn_s and Attn_c indicate the self-attention and cross-attention layers, respectively. A causal attention mask can be used to compute S^ℓ in parallel, as in Figure 1a.

B Computation for $\tilde{h}_k^\ell$ in BERT

Assuming the model consists of L layers. Without the cross-attention, $\tilde{h}_k^\ell$ is the feed-forward result of h_k^ℓ:

$$h_k^\ell = \mathrm{Attn}_s(\tilde{h}_k^{\ell-1}, \widetilde{H}_{\backslash\kappa}^{\ell-1}, \widetilde{H}_{\backslash\kappa}^{\ell-1}), \quad (14)$$

where $\tilde{h}_k^0$ is the embedding of the k-th token in $\boldsymbol{x}_{\backslash\kappa}$ and $\widetilde{H}_{\backslash\kappa}^{\ell-1} = (\tilde{h}_1^{\ell-1}, \ldots, \tilde{h}_m^{\ell-1})$ denotes a set of hidden states for $\boldsymbol{x}_{\backslash\kappa}$. Compared with s_j^ℓ, h_k^ℓ utilizes both the left and right sides of the context of the masked token x_k to capture deeper representations.

C Procedure for Prediction

Figure 5 shows our procedure for prediction.

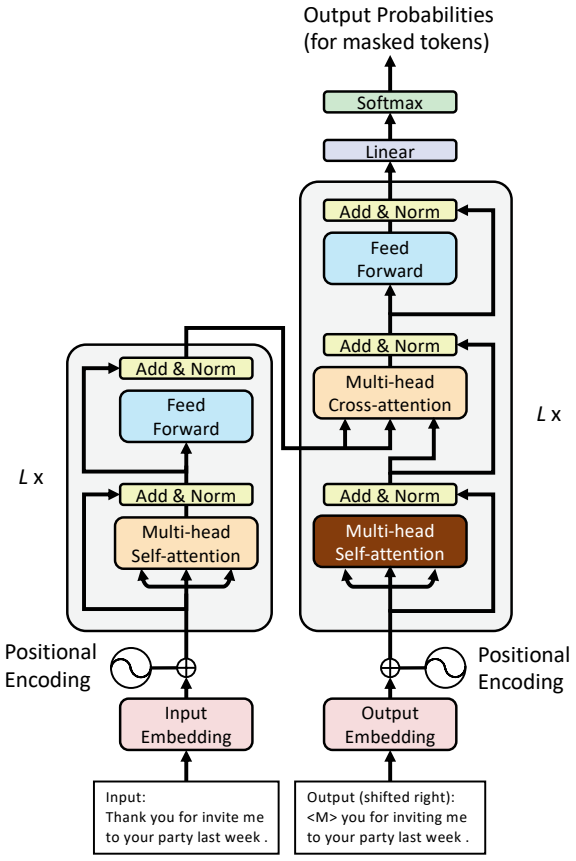


Figure 5: Bidirectional Transformer architecture. The left and right columns indicate the encoder and decoder, respectively. The self-attention mechanism in the decoder utilizes the fully-visible mask (Figure 1b), unlike the conventional Transformer (Vaswani et al., 2017).

D Pre-training Loss

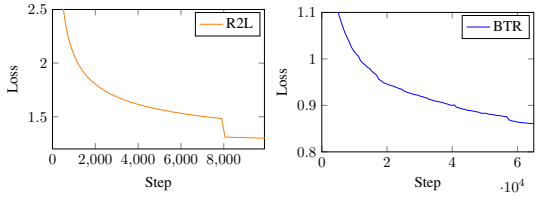


Figure 6: Pre-training loss for R2L (left) and the BTR (right).

Figure 6 shows the pre-training loss for R2L and the BTR on the Realnewslike corpus. The training loss of R2L suddenly dropped from 1.48 to 1.3 after the first epoch (7957 steps).

E Cleaning for CoNLL Corpus

The original texts of CoNLL-13 and 14 contain several styles of punctuation tokenization, such as "DementiaToday,2012" and "known , a". While these

Model	Precision	Recall	$F_{0.5}$
Oracle	80.62	51.98	72.62
T5GEC	78.01	48.57	69.58
R2L	**78.81**	46.83	69.34
w/o L2R	77.69	46.55	68.52
BERT	58.84	**53.53**	57.70
RoBERTa ($\lambda = 0.1$) w/o a_{train}	77.86	48.62	69.50
w/o a_{train}, λ	71.07	47.36	64.60
BTR ($\lambda = 0.4$)	78.52	48.82	**70.00**
w/o λ	76.02	48.30	68.19
w/o a_{train}, λ	67.44	49.45	62.87

Table 8: Results for the models on the cleaned CoNLL-14 corpus with candidates from T5GEC. Bold scores represent the highest precision, recall, and $F_{0.5}$.

Model	$F_{0.5}$
R2L	69.36 ± 0.13
RoBERTa ($\lambda = 0.1$) w/o a_{train}	68.12 ± 2.39
BTR ($\lambda = 0.4$)	**69.80** ± 0.18

Table 9: The mean ± std results on the cleaned CoNLL-14 corpus with candidates from T5GEC. Bold scores represents the highest mean.

punctuation styles with/without spaces are not considered grammatical errors by a human, they are often identified as errors by automatic GEC scorers. Moreover, while most of the sequences in CoNLL-14 are of sentence-level, several sequences are of paragraph-level due to the punctuation without spaces. In this research, we cleaned the texts of CoNLL-13 and 14 using the "en_core_web_sm" tool in spaCy (Honnibal et al., 2020) so that all punctuation included spaces. The paragraph-level sequences were split into sentences with respect to the position of full stops. The cleaned CoNLL-14 corpus contains 1326 pairs of data.

Tables 8, 9, and 10 show the experimental results on the cleaned CoNLL-14 corpus.

F Hyperparameters

Table 11 lists the hyperparameter settings used for each model. And Table 12 lists the used artifacts. The setting for T5GEC (large) was the same as T5GEC. We followed the setting of Kaneko et al. (2019) to use a 0.0005 learning rate for the BERT reranker. We used a 0.0001 learning rate for the RoBERTa reranker. For both BERT and RoBERTa, we utilized the adam optimizer, "inverse square root" learning rate schedule, and 1.2 epochs warm-up steps. For other models based on a T5 structure, we used a 0.001 learning rate and adafactor optimizer. The batch size was 1048576 tokens for all models. We used the Fairseq (Ott et al., 2019) and HuggingFace (Wolf et al., 2020) to reproduce all models and run the BTR.

Model	Precision	Recall	$F_{0.5}$
Oracle	82.01	54.19	74.38
T5GEC (large)	79.27	49.91	70.92
R2L	**79.72**	48.71	70.72
RoBERTa ($\lambda = 0.1$) w/o a_{train}	79.30	49.91	70.94
BTR ($\lambda = 0.8$)	79.65	**49.98**	**71.20**

Table 10: Results for the models on the cleaned CoNLL-14 corpus with candidates from T5GEC (large). Bold scores represent the highest precision, recall, and $F_{0.5}$.

G Candidate and Threshold Tuning

Following Zhang et al. (2021), we tuned a for training and predicting separately on the validation dataset with candidates generated by T5GEC. Table 13 lists the size of training data with candidates generated by T5GEC. When tuning $a_{train} \in \{0, 5, 10, 20\}$[5] for the BTR, a_{pred} was fixed to 5. Because the BTR with $\lambda = 0.4$ and $a_{train} = 20$ achieved the highest score as shown in Table 14, a_{train} was fixed to 20, this BTR was also used to tune $a_{pred} \in \{5, 10, 15, 20\}$. When tuning $a_{train} \in \{0, 5, 10, 20\}$ for RoBERTa, a_{pred} was fixed to 5. The results in Tables 14 and 15 indicate the different distributions of $F_{0.5}$ score between RoBERTa and the BTR. To investigate the reason, we compared the training loss and $F_{0.5}$ score of RoBERTa with the BTR. Figure 7 shows the comparison. Different from the BTR, when using negative sampling ($a_{train} > 0$) for training RoBERTa, the $F_{0.5}$ score on the CoNLL-13 corpus decreased with the epoch increasing. The training loss of RoBERTa also dropped suddenly after finishing the first epoch. This result suggests that negative sampling in the GEC task for an encoder-only structure leads in the wrong direction in learning representations from the concatenated source and target. And therefore, we fixed a_{train} to 0 for RoBERTa. This RoBERTa was also used to tune $a_{pred} \in \{5, 10, 15, 20\}$. The results in Tables 16, 17, and 18 show that when a_{pred} was set to 5, the BTR, R2L, RoBERTa, and BERT attained their highest scores on the CoNLL-13 corpus. Thus, a_{pred} was fixed to 5 in our experiments.

Tables 14 and 16 also show the performances

[5] Setting a to 0 indicates training with only gold data.

Hyperparameters	T5GEC	BERT	RoBERTa	R2L (pretrain)	R2L (finetune)	BTR (pretrain)	BTR (finetune)
# of updates	15 (epochs)	15 (epochs)	15 (epochs)	10000	15 (epochs)	65536	15 (epochs)
Max src / tgt length (train)	128	128	128	512	128	512	128
Max src / tgt length (eval)	512	1	512	512	512	512	512
a_{train}	-	-	{0, 5, 10, 20}	-	-	-	{0, 5, 10, 20}
a_{pred}	-	{5, 10, 15, 20}	{5, 10, 15, 20}	-	{5, 10, 15, 20}	-	{5, 10, 15, 20}
Threshold (λ)	-	-	{0, 0.1, 0.2,. . . ,0.9}	-	-	-	{0, 0.1, 0.2,. . . ,0.9}

Table 11: Used hyperparameters.

Used artifacts	Note
T5-base	https://huggingface.co/google/t5-v1_1-base
T5-large	https://huggingface.co/google/t5-v1_1-large
T5GEC	https://github.com/google-research-datasets/clang8/issues/3
RoBERTa	https://huggingface.co/roberta-base
BERT	https://huggingface.co/bert-large-cased
cLang-8	https://github.com/google-research-datasets/clang8
CoNLL-13	File *revised/data/official-preprocessed.m2*
CoNLL-14	File *alt/official-2014.combined-withalt.m2*
JFLEG	File *test/test.src*
ERRANT	https://github.com/chrisjbryant/errant
Fairseq	https://github.com/facebookresearch/fairseq/
HuggingFace	https://github.com/huggingface/transformers/
BEA-19 competition	https://competitions.codalab.org/competitions/20229

Table 12: Used artifacts.

a_{train}	# of training data (pairs)
0	2,371,961
5	13,727,133
10	22,396,187
20	30,423,347

Table 13: Number of sentence pairs for cLang-8 dataset with candidates. All pairs of data that satisfy the length constraint of 128 are listed.

of the BTR concerning λ on the CoNLL-13 corpus with candidates generated by T5GEC. Without using any candidate for training, the BTR($\lambda = 0$) could achieve the highest $F_{0.5}$ score. When using 20 candidates for training, the BTR ($\lambda = 0.4$) achieved the highest $F_{0.5}$ score of 50.22. Table 19 shows the BTR ($a_{train} = 20, \lambda = 0.8$) achieved the highest $F_{0.5}$ score on the CoNLL-13 dataset with the candidates generated by T5GEC(large). Thus, our tuned λ for the BTR was set to 0.2 when $a_{train} = 0$. When $a_{train} = 20$, λ was set to 0.4 and 0.8 for the candidates generated by T5GEC and T5GEC(large), respectively. Similarly, when $a_{train} = 0$, our tuned λ for RoBERTa was set to 0.1 for the two versions of candidates.

H Mean and Standard Deviation

We list the mean and standard deviation of R2L, RoBERTa, and the BTR over the four trails on each dataset in Table 20.

I Detailed Results on BEA Test

The distribution of the BEA test set with respect to the CEFR level is shown in Table 21.

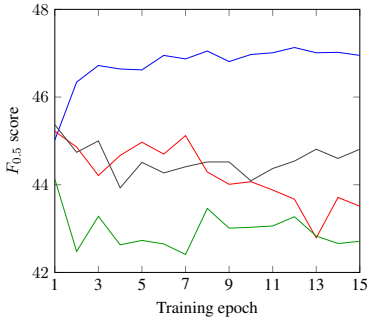
The BTR ($\lambda = 0.4$) achieved an $F_{0.5}$ score of 71.27 on the BEA test set, as shown in Table 22. Compared with A (beginner) level sentences, the BTR was more effective for B (intermediate), C (advanced), and N (native) level sentences. As shown in Table 23, the BTR ($\lambda = 0.4$) improved T5GEC for all top-5 error types. Furthermore, the BTR ($\lambda = 0.4$) could effectively handle *Missing* and *Unnecessary* tokens but not *Replacement* for the native sentences. It was more difficult to correct the *Replacement* and *Unnecessary* operations in the native sentences for both models compared with the advanced sentences. This may be because the writing style of native speakers is more natural and difficult to correct with limited training data, whereas language learners may tend to use a formal language to make the correction easier.

J Relation Between a, λ, and BTR Performance

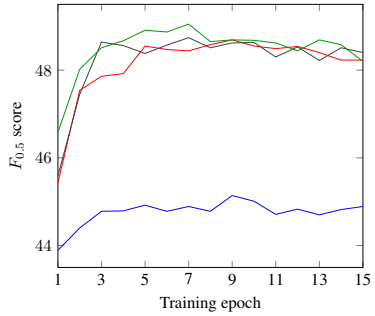
The BEA and JFLEG corpus also provide a dev set with 4384 and 754 sentences for validation, respectively. To determine the optimal a_{train}, a_{pred}, and λ for the BTR listed in Table 14 on these two datasets, we re-evaluated the performances of the

$F_{0.5}$ / a_{train} \ **Threshold**(λ)	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0	**45.34**					49.36					
5	49.14	49.10	49.64	49.86	**49.92**	49.87	49.61	49.19	49.37	49.36	
10	48.84	49.50	49.62	50.09	**50.10**	50.07	49.96	49.91	49.91	49.57	49.36
20	49.13	49.42	49.74	50.08	**50.22**	49.89	50.00	49.92	49.62	49.46	49.36

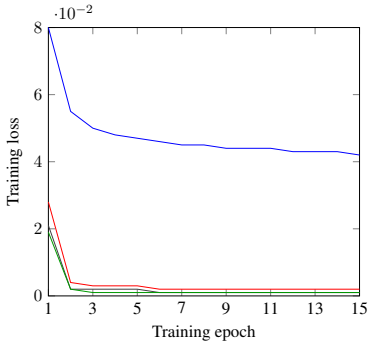
Table 14: Results of tuning a_{train} for BTR. a_{pred} was fixed to 5. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each a_{train} among different threshold is shown in bold. The scores that were the same as those of the base model ($\lambda = 1$) were ignored and greyed out.



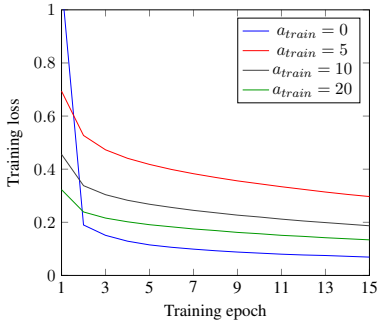
(a) $F_{0.5}$ score of RoBERTa on the CoNLL-13 corpus



(b) $F_{0.5}$ score of BTR on the CoNLL-13 corpus



(c) Training loss of RoBERTa



(d) Training loss of BTR

Figure 7: Performances of BTR and RoBERTa with various a_{train} without λ during fine-tuning. a_{pred} was fixed to 5 with candidates from T5GEC. Both $F_{0.5}$ score and training loss were averaged over the four trials.

$F_{0.5}$ / a_{train} \ **Threshold**(λ)	0	0.1	0.2, . . . , 1
0	46.48	**49.35**	49.36
5	44.89	**49.38**	49.36
10	45.68	**49.38**	49.36
20	41.91	**49.38**	49.36

Table 15: Results of tuning a_{train} for RoBERTa. a_{pred} was fixed to 5. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each a_{train} among different threshold is shown in bold. The scores that were the same as those of the base model ($\lambda = 1$) were ignored and greyed out.

BTR on the corresponding dev sets. Tables 24 and 25 show the results on the BEA and JFLEG dev sets, respectively. On the BEA dev set, the highest $F_{0.5}$ score of 54.04 was achieved with $a_{train} = 10$, $a_{pred} = 5$, and $\lambda = 0.2$. On the JFLEG dev set, the highest GLEU score of 54.46 was achieved with $a_{train} = 5$, $a_{pred} = 15$, and $\lambda = 0$. These results demonstrate the differences in evaluating the minimal edit and fluency for grammar corrections. Given the previous a_{train}, a_{pred} and λ, we re-evaluated the BTR on the BEA and JFLEG test sets. Table 26 lists the results. Tuning hyperparameters on the JFLEG dev set led to a higher GLEU score of 60.14 on the JFLEG test set, compared to the tuned hyperparameters on the CoNLL-13 set. However, tuning hyperparameters on the BEA dev set resulted in a lower $F_{0.5}$ score of 71.12 on the BEA test set, compared to the tuned hyperparameters on the CoNLL-13 set.

To investigate the effectiveness of λ, i.e., the parameter that balances the trade-off between accep-

$F_{0.5}$ Threshold(λ) / a_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
5	49.13	49.42	49.74	50.08	**50.22**	49.89	50.00	49.92	49.62	49.46	49.36
10	48.92	49.34	**50.01**	49.85	49.85	49.51	49.71	49.62	49.49	49.39	49.40
15	48.91	49.22	**49.65**	49.36	49.21	49.18	49.04	49.08	48.90	48.92	48.88
20	36.50	36.83	38.21	38.85	40.24	41.84	43.11	44.41	45.65	**46.87**	49.40

Table 16: Results of tuning a_{pred} for BTR. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each a_{pred} among different threshold is shown in bold. The scores that were same as those of the base model ($\lambda = 1$) were ignored and greyed out.

$F_{0.5}$ Threshold(λ) / a_{pred}	0	0.1	0.2, . . . , 1
5	46.48	**49.35**	49.36
10	**46.08**		49.40
15	**45.04**		48.88
20	**44.28**		49.40

Table 17: Results of tuning a_{pred} for RoBERTa. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each a_{pred} among different threshold is shown in bold. The scores that were same as those of the base model ($\lambda = 1$) were ignored and greyed out.

Dataset	a_{pred}	R2L	BERT
CoNLL-13	5	**50.02**	**42.44**
	10	49.94	40.53
	15	49.85	39.98
	20	39.81	39.37

Table 18: Results of tuning a_{pred} for R2L and BERT. The highest $F_{0.5}$ score on the CoNLL-13 corpus for each reranker among different a_{pred} is shown in bold.

tance rate and quality of grammatical corrections, we analyzed the relationship between λ and the corresponding precision, recall, and GLEU scores. Figures 8 and 9 show the performance of the BTR ($a_{train} = 20, a_{pred} = 5$) on the CoNLL-13 and 14 corpus, respectively. With λ increasing, the acceptance rate, i.e., the percentage of suggestions that the BTR accepts, decreases while the precision and recall for the *Accept* suggestions increases. This demonstrates our assumption in Section 4.3 that the value of $f(\boldsymbol{y}|\boldsymbol{x})$ indicates the confidence of the BTR, and the confident suggestions tended to be more grammatical, while the unconfident ones tended to be less grammatical than the original selections. As for the whole corpus, when $\lambda = 0.7$, this BTR achieved lower precision and recall score than $\lambda = 0.4$ due to the limited amount of *Accept* suggestions. Figures 10 and 11 show the performance of BTR ($a_{train} = 10, a_{pred} = 5$) on the BEA dev and test corpus, respectively. In Figure 10, the BTR shows a similar performance to that on the CoNLL-13 and 14 that, where a larger λ leads to higher precision and recall for *Accept* suggestions. However, the performance over the whole corpus also depends on the acceptance rate. Differently, as shown in Figures 13 and 14, the experimental results of the BTR ($a_{train} = 5, a_{pred} = 15$) on the JFLEG corpus achieved the highest GLEU score for the whole corpus when $\lambda \leq 0.1$. This may be because using $a_{pred} = 15$ makes a flatter probability than $a_{pred} = 5$ as shown in Figures 12 and 15. Besides, recognizing the fluency of a sentence by the BTR may be easier than recognizing the minimal edit of corrections.

K Precision and Recall With T5GEC (large) Candidates

Given the top-5 candidate sentences generated by T5GEC (large), we compared the precision and recall of the BTR with those of R2L and RoBERTa in Table 27.

L Example of Reranked Outputs

Table 28 provides examples of ranked outputs by T5GEC, R2L, RoBERTa w/o a_{train} ($\lambda = 0.1$), and BTR ($\lambda = 0.4$). The first block of output results demonstrates the difficulty of correcting spelling errors. In this block, the BTR outputs the token "insensitively" with the correct spelling but a mismatched meaning, whereas other rerankers tend to keep the original token "intesively" with a spelling error. The examples in the second block show that both the BTR and R2L are capable of correctly addressing verb tense errors. The examples in the last block show that even though the BTR recognizes the missing determiner "the" for the word "Disadvantage", it still misses a that-clause sentence.

M Inference Time Cost

In inference, we required all rerankers to compute one target sequence at a time to estimate the time

Model	λ										
	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
RoBERTa	47.90	**50.76**					50.79				
BTR	49.44	50.17	50.00	49.98	49.98	50.58	50.47	50.92	**51.00**	50.82	50.79

Table 19: Results of RoBERTa and BTR on the CoNLL-13 corpus with candidates generated by T5GEC (large). The scores that were the same as those of the base model ($\lambda = 1$) were ignored and greyed out.

Model	CoNLL-13	CoNLL-14	BEA	JFLEG
R2L	50.02 ± 0.15	64.96 ± 0.10	**71.42** ± 0.05	59.09 ± 0.19
RoBERTa w/o a_{train}	48.66 ± 1.20	63.96 ± 1.90	68.90 ± 2.88	58.68 ± 0.66
BTR	**50.05** ± 0.07	**65.29** ± 0.19	70.84 ± 0.05	**59.12** ± 0.07

Table 20: The mean $\pm$ std results on each dataset with candidates from T5GEC. Bold scores are the highest mean for each dataset.

Dataset	Level	# of data (pairs)
BEA	A	1,107
	B	1,330
	C	1,010
	N	1,030

Table 21: Dataset size of the BEA test. Each sentence in the BEA test set is classified into either A (beginner), B (intermediate), C (advanced), or N (native) corresponding to the CEFR level.

cost. For RoBERTa and the BTR, we rearranged the given target sequence by masking each token. These rearranged sequences were then put into a mini-batch for parallel computation. For T5GEC, given the source sentence, we used the mini-batch with a size of 5 to parallelly compute all beams.

Table 29 displays the time cost for each model to estimate scores over the entire corpus with 5 candidates, using one NVIDIA A100 80GB GPU. We only calculated the time for estimating probability and ignored the time for loading the model and dataset. T5GEC costs the most time among all rerankers, as it predicts tokens of the target sequence one by one. RoBERTa and the BTR took longer than BERT and R2L due to the target sequence rearrangement procedure. The BTR took 2 to 3 times as much as RoBERTa due to the additional decoder structure.

Model	Level	Missing	Replacement	Unnecessary	All
T5GEC	A	62.30	**69.92**	73.74	68.40
	B	73.99	67.94	78.26	70.93
	C	78.54	71.51	85.16	75.54
	N	80.66	**69.48**	53.78	71.36
	All	71.23	69.47	74.30	70.51
BTR ($\lambda = 0.4$)	A	**63.65**	69.86	**74.40**	**68.76**
	B	**74.81**	**68.94**	**79.01**	**71.84**
	C	**81.85**	**72.61**	**86.00**	**77.36**
	N	**83.17**	68.23	**57.88**	**72.01**
	all	**72.91**	**69.69**	**75.69**	**71.27**

Table 22: Results for each operation type with classified CEFR levels on the BEA test set with candidates from T5GEC. Edit operations are divided into *Missing*, *Replacement*, and *Unnecessary* corresponding to inserting, substituting, and deleting tokens, respectively. Bold scores are the highest for each operation with the corresponding level.

Model	PUNCT	DET	PREP	ORTH	SPELL
T5GEC	74.62	77.57	73.33	70.32	78.38
BTR ($\lambda = 0.4$)	**75.73**	**79.08**	**73.77**	**70.72**	**78.87**

Table 23: Results for the top five error types on the BEA test set. Bold scores are the highest for each error type.

$F_{0.5}$ **Threshold**(λ) / a_{train}, a_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0, 5	47.44	**52.83**	52.77	52.52				52.51			
0, 10	44.93	**52.65**					52.37				
0, 15	43.74	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	**52.46**	52.45	
0, 20	31.01	51.86	52.21	52.33	52.38	52.41	52.42	52.44	**52.45**	**52.45**	52.47
5, 5	52.51	53.22	**53.49**	53.41	53.42	53.45	53.40	53.28	53.17	53.06	52.51
5, 10	51.21	53.19	53.52	53.41	53.40	**53.56**	53.34	53.21	53.15	53.04	52.37
5, 15	50.68	53.11	53.37	53.45	**53.54**	53.46	53.30	53.23	53.20	53.10	52.45
5, 20	30.44	32.42	34.86	36.39	38.28	41.33	42.73	43.90	45.16	**46.67**	52.47
10, 5	53.47	53.95	**54.04**	53.95	53.85	53.68	53.64	53.38	53.09	53.01	52.51
10,10	52.51	53.21	**53.99**	53.87	53.70	53.73	54.49	53.30	53.02	52.99	52.37
10,15	52.05	53.97	**54.01**	53.64	53.66	53.63	53.44	53.26	53.03	53.05	52.45
10,20	30.03	31.55	32.76	34.12	36.07	39.25	40.89	42.50	43.68	**45.45**	52.47
20, 5	53.26	**53.87**	53.85	53.75	53.79	53.77	53.70	53.50	53.31	52.98	52.51
20,10	52.37	53.15	53.75	53.77	**53.91**	53.83	53.54	53.44	53.24	53.15	52.37
20,15	52.29	53.69	53.82	**53.85**	53.84	53.64	53.46	53.33	53.21	53.21	52.45
20,20	29.68	31.03	32.11	33.47	35.15	38.52	39.99	41.64	43.25	**44.95**	52.47

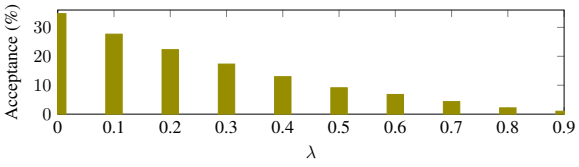
Table 24: Results of tuning a_{train} and a_{pred} for BTR on the BEA dev set. The highest $F_{0.5}$ score for each pair of a_{train} and a_{pred} among different threshold is shown in bold. The scores that were the same as those of the base model ($\lambda = 1$) were ignored and greyed out.

GLEU **Threshold**(λ) / a_{train}, a_{pred}	0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1
0, 5	**52.43**						53.25				
0, 10	**51.91**						53.25				
0, 15	**51.36**						53.25				
0, 20	44.59	**52.97**					53.25				
5, 5	54.35	**54.37**	54.12	54.05	53.81	53.67	53.42	53.32	53.20	53.22	53.25
5, 10	**54.41**	54.34	54.05	53.68	53.48	53.31	53.30	53.26	53.26	53.25	
5, 15	**54.46**	54.44	53.88	53.43	53.33	53.29	53.22	53.22	53.26	53.20	53.25
5, 20	44.42	44.99	45.82	46.32	46.80	47.43	47.73	48.13	48.98	**49.37**	53.25
10, 5	54.15	**54.23**	53.99	53.88	53.69	53.51	53.37	53.28	53.24	53.23	53.25
10,10	**54.23**	54.20	53.93	53.73	53.54	53.37	53.29	53.24	53.24	53.23	53.25
10,15	**54.29**	54.16	53.89	53.57	53.48	53.33	53.26	53.19	53.22	53.23	53.25
10,20	44.22	44.71	45.29	45.70	46.20	47.21	47.45	47.98	48.52	**49.07**	53.25
20, 5	**53.92**	53.87	53.85	53.68	53.60	53.49	53.50	53.38	53.25	53.26	53.25
20,10	**53.92**	53.88	53.65	53.54	53.53	53.42	53.32	53.22	53.23	53.26	53.25
20,15	**54.12**	53.89	53.61	53.42	53.42	53.38	53.28	53.23	53.19	53.22	53.25
20,20	44.37	44.79	45.22	45.56	45.98	46.93	47.36	47.83	48.48	**49.08**	53.25

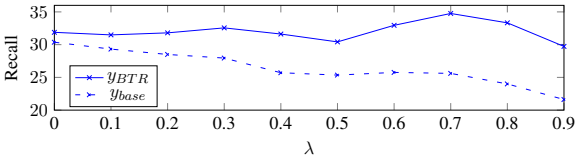
Table 25: Results of tuning a_{train} and a_{pred} for BTR on the JFLEG dev set. The highest GLEU score for each pair of a_{train} and a_{pred} among different threshold is shown in bold. The scores that were the same as those of the base model ($\lambda = 1$) were ignored and greyed out.

Tuned on corpus	a_{train}	a_{pred}	λ	BEA	JFLEG
CoNLL-13	20	5	0.4	**71.27**	59.17
BEA dev	10	5	0.2	71.12	-
JFLEG dev	5	15	0	-	**60.14**

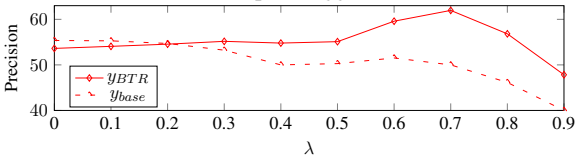
Table 26: Results for BTR on the BEA and JFLEG test sets with tuned hyperparameters.



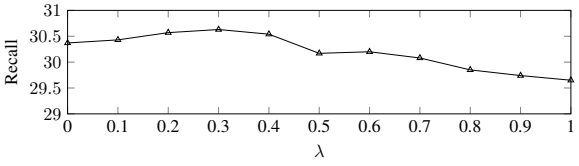
(a) Accept rate versus λ



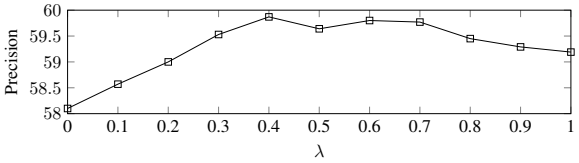
(b) Recall for accepted suggestions versus λ



(c) Precision for accepted suggestions versus λ

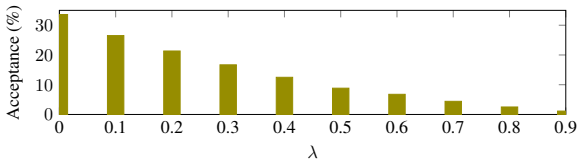


(d) Recall over the whole corpus versus λ

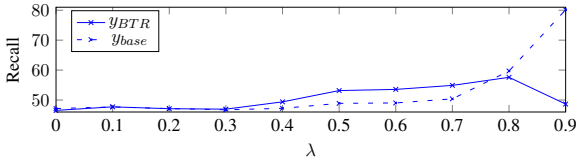


(e) Precision over the whole corpus versus λ

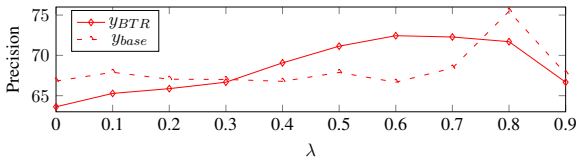
Figure 8: Precision and recall of BTR ($a_{train} = 20, a_{pred} = 5$) with respect to different λ on the CoNLL-13 set.



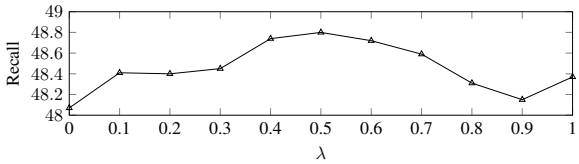
(a) Accept rate versus λ



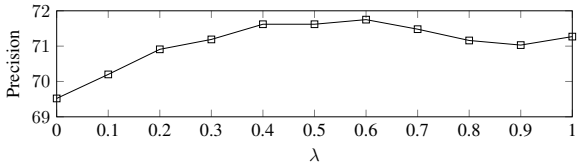
(b) Recall for accepted suggestions versus λ



(c) Precision for accepted suggestions versus λ



(d) Recall over the whole corpus versus λ



(e) Precision over the whole corpus versus λ

Figure 9: Precision and recall of BTR ($a_{train} = 20, a_{pred} = 5$) with respect to different λ on the CoNLL-14 set.

Model	CoNLL-13		CoNLL-14		BEA test	
	p	r	p	r	p	r
Oracle	66.34	35.34	76.04	53.36	-	-
T5GEC (large)	60.24	31.20	73.10	**49.76**	75.65	60.87
R2L	**61.55**	30.03	**73.60**	48.47	**77.06**	60.24
RoBERTa ($\lambda = 0.1$) w/o a_{train}	60.22	31.17	73.12	**49.76**	75.74	60.83
BTR ($\lambda = 0.8$)	60.54	**31.28**	72.71	**49.76**	75.91	**61.13**

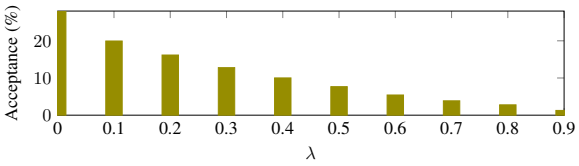
Table 27: The (p)recision and (r)ecall on each dataset. The top-5 candidate sentences were generated by T5GEC (large). Bold scores represent the highest precision and recall for each dataset.

Source	However , it is a good practice not to intesively use social media all the time .
Gold 1	However , it is a good practice not to intensely use social media all the time .
Gold 2	However , it is good practice not to intensively use social media all the time .
Candidate 1 (R2L, RoBERTa, T5GEC)	However , it is good practice not to intesively use social media all the time .
Candidate 2	However , it is good practice not to intensely use social media all the time .
Candidate 3 (BTR)	However , it is good practice not to insensitively use social media all the time .
Source	It is true that social media makes people be able to connect one another more conveniently .
Gold 1	It is true that social media allows people to connect to one another more conveniently .
Gold 2	It is true that social media make people able to connect with one another more conveniently
Candidate 1 (RoBERTa, T5GEC)	It is true that social media makes people be able to connect with one another more conveniently .
Candidate 2 (BTR, R2L)	It is true that social media makes people able to connect with one another more conveniently .
Candidate 3	It is true that social media makes people able to connect to one another more conveniently .
Source	Disadvantage is parking their car is very difficult .
Gold 1	A disadvantage is that parking their cars is very difficult .
Gold 2	A disadvantage is that parking their car is very difficult .
Gold 3	The disadvantage is that parking their car is very difficult .
Candidate 1 (R2L, RoBERTa, T5GEC)	Disadvantage is parking their car is very difficult .
Candidate 2 (BTR)	The disadvantage is parking their car is very difficult .
Candidate 3	The disadvantage is that parking their car is very difficult .

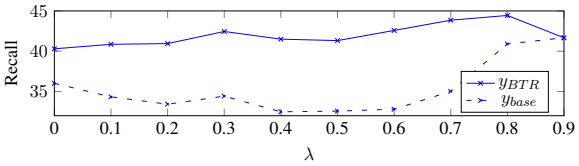
Table 28: Examples of reranked outputs. The 3 candidate sentences were generated by T5GEC. Blue indicates the range of corrections. Examples in the first two and last block were extracted from the CoNLL-14 and JFLEG test corpus, respectively.

Model	CoNLL-13	CoNLL-14	BEA dev	BEA test	JFLEG dev	JFLEG test
T5GEC	778	790	3638	3776	451	444
BERT	22	21	68	69	12	13
R2L	34	32	108	109	19	19
RoBERTa	82	88	333	386	46	69
BTR	194	199	740	738	113	122

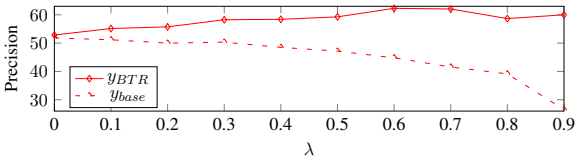
Table 29: Time cost (seconds) in inference over the whole corpus with 5 candidates generated by T5GEC.



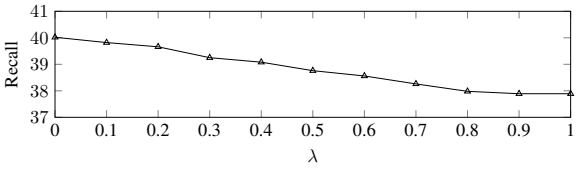
(a) Accept rate versus λ



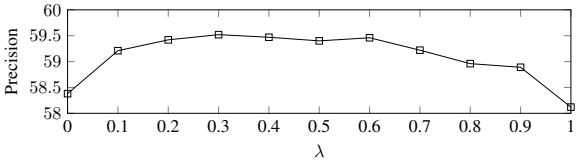
(b) Recall for accepted suggestions versus λ



(c) Precision for accepted suggestions versus λ

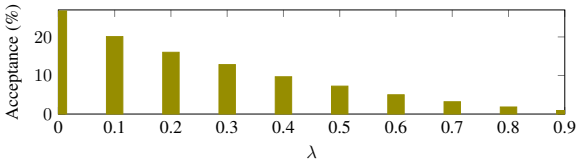


(d) Recall over the whole corpus versus λ

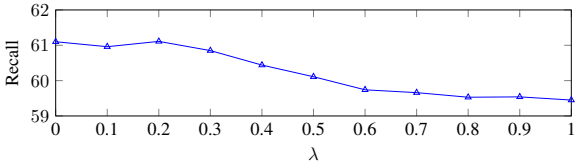


(e) Precision over the whole corpus versus λ

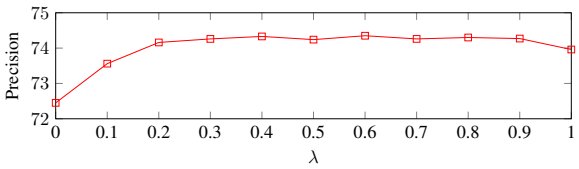
Figure 10: Precision and recall of BTR ($a_{train} = 10, a_{pred} = 5$) with respect to different λ on the BEA dev set.



(a) Accept rate versus λ



(b) Recall over the whole corpus versus λ



(c) Precision over the whole corpus versus λ

Figure 11: Precision and recall of BTR ($a_{train} = 10, a_{pred} = 5$) with respect to different λ on the BEA test set.

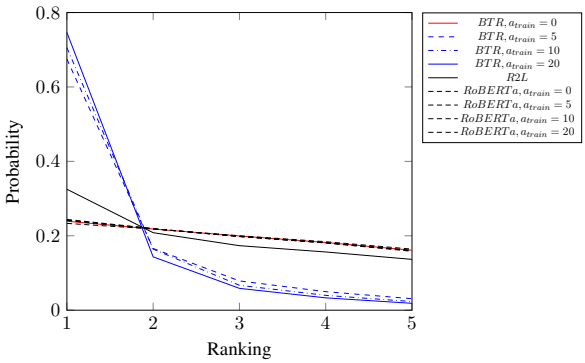
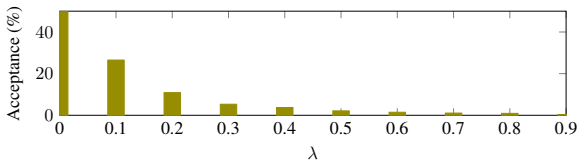
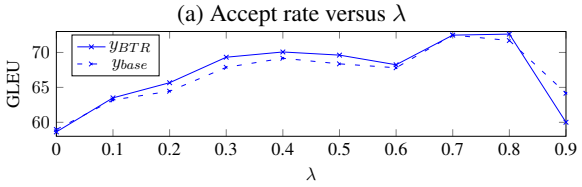


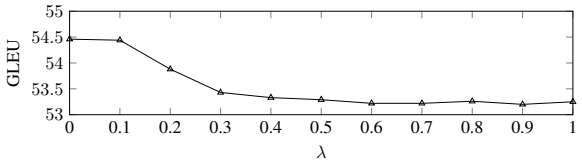
Figure 12: Average probability for each rank on the BEA test set. The top-5 candidate sentences were generated by T5GEC.



(a) Accept rate versus λ

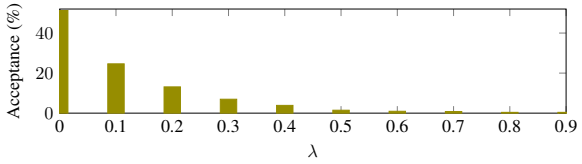


(b) GLEU for accepted suggestions versus λ

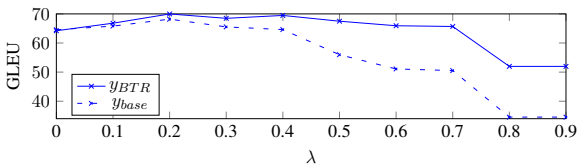


(c) GLEU over the whole corpus versus λ

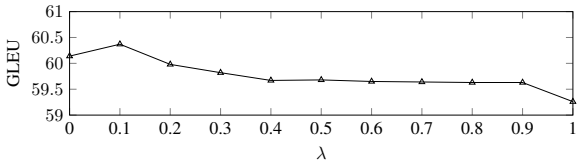
Figure 13: GLEU of BTR ($a_{train} = 5, a_{pred} = 15$) with respect to different λ on the JFLEG dev set.



(a) Accept rate versus λ



(b) GLEU for accepted suggestions versus λ



(c) GLEU over the whole corpus versus λ

Figure 14: GLEU of BTR ($a_{train} = 5, a_{pred} = 15$) with respect to different λ on the JFLEG test set.

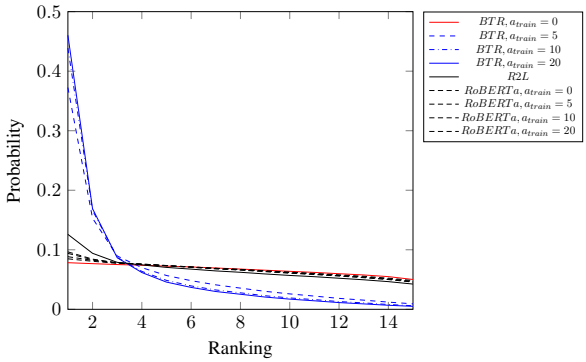


Figure 15: Average probability for each rank on the JFLEG test set. The top-15 candidate sentences were generated by T5GEC.

ACL 2023 Responsible NLP Checklist

A For every submission:

☑ A1. Did you describe the limitations of your work?
Section 7

☒ A2. Did you discuss any potential risks of your work?
This research is a foundational research and not tied to risk applications. The artifacts we used are all open-source and widely used for the research purpose.

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

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Abstract, Section 3.2, Section 5, Appendix E, and Appendix F.

☑ B1. Did you cite the creators of artifacts you used?
Section 3.2, Section 5, Appendix E, and Appendix F.

☒ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
For each artifact we used, we included its URL or citation which contained the license.

☒ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
We used the existing artifacts only for this research purpose and there was no usage toward the commerce in our paper. As for the artifacts we create, we will clarify the usage of our artifacts in our Github.

☒ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
The datasets we used were open-source or automatically generated by an artifacts, so there is no anonymization problem.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
Section 5.3

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
Section 5.3 and Appendix G

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

C ☑ Did you run computational experiments?

Appendix M

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
Section 5.1, Section 5.2, and Appendix M

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
Section 5.2, Section 5.5, Appendix G, and Appendix J

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
Section 5.1, Section 5.2, and Appendix M

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Section 3.2, Section 5, Appendix E, and Appendix F.

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

Left blank.

☐ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
No response.

☐ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
No response.

☐ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
No response.

☐ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
No response.

☐ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
No response.