# An Empirical Study of Incorporating Pseudo Data into Grammatical Error Correction

**Shun Kiyono** [1,2] **Jun Suzuki** [2,1] **Masato Mita** [1,2] **Tomoya Mizumoto** [1,2*] **Kentaro Inui** [2,1]
[1] RIKEN Center for Advanced Intelligence Project [2] Tohoku University
{shun.kiyono, masato.mita, tomoya.mizumoto}@riken.jp;
{jun.suzuki,inui}@ecei.tohoku.ac.jp



## Abstract

The incorporation of pseudo data in the training of grammatical error correction models has been one of the main factors in improving the performance of such models. However, consensus is lacking on experimental configurations, namely, choosing how the pseudo data should be generated or used. In this study, these choices are investigated through extensive experiments, and state-of-the-art performance is achieved on the CoNLL-2014 test set ($F_{0.5} = 65.0$) and the official test set of the BEA-2019 shared task ($F_{0.5} = 70.2$) without making any modifications to the model architecture.

## 1 Introduction

To date, many studies have tackled grammatical error correction (GEC) as a machine translation (MT) task, in which ungrammatical sentences are regarded as the source language and grammatical sentences are regarded as the target language. This approach allows cutting-edge neural MT models to be adopted. For example, the encoder-decoder (EncDec) model (Sutskever et al., 2014; Bahdanau et al., 2015), which was originally proposed for MT, has been applied widely to GEC and has achieved remarkable results in the GEC research field (Ji et al., 2017; Chollampatt and Ng, 2018; Junczys-Dowmunt et al., 2018).

However, a challenge in applying EncDec to GEC is that EncDec requires a large amount of training data (Koehn and Knowles, 2017), but the largest set of publicly available parallel data (Lang-8) in GEC has only two million sentence pairs (Mizumoto et al., 2011). Consequently, the method of augmenting the data by incorporating pseudo training data has been studied intensively (Xie et al., 2018; Ge et al., 2018; Lichtarge et al., 2019; Zhao et al., 2019).

When incorporating pseudo data, several decisions must be made about the experimental configurations, namely, (i) the method of generating the pseudo data, (ii) the seed corpus for the pseudo data, and (iii) the optimization setting (Section 2). However, consensus on these decisions in the GEC research field is yet to be formulated. For example, Xie et al. (2018) found that a variant of the backtranslation (Sennrich et al., 2016b) method (BACKTRANS (NOISY)) outperforms the generation of pseudo data from raw grammatical sentences (DIRECTNOISE). By contrast, the current state of the art model (Zhao et al., 2019) uses the DIRECTNOISE method.

In this study, we investigate these decisions regarding pseudo data, our goal being to provide the research community with an improved understanding of the incorporation of pseudo data. Through extensive experiments, we determine suitable settings for GEC. We justify the reliability of the proposed settings by demonstrating their strong performance on benchmark datasets. Specifically, without any task-specific techniques or architecture, our model outperforms not only all previous single-model results but also all ensemble results except for the ensemble result by Grundkiewicz et al. (2019)[1]. By applying task-specific techniques, we further improve the performance and achieve state-of-the-art performance on the CoNLL-2014 test set and the official test set of the BEA-2019 shared task.

## 2 Problem Formulation and Notation

In this section, we formally define the GEC task discussed in this paper. Let $\mathcal{D}$ be the GEC training data that comprise pairs of an ungrammatical source sentence $\boldsymbol{X}$ and grammatical target sentence

[*] Current affiliation: Future Corporation

[1] The paper (Grundkiewicz et al. 2019) has not been published yet at the time of submission.

$\boldsymbol{Y}$, i.e., $\mathcal{D} = \{(\boldsymbol{X}_n, \boldsymbol{Y}_n)\}_n$. Here, $|\mathcal{D}|$ denotes the number of sentence pairs in the dataset $\mathcal{D}$.

Let $\boldsymbol{\Theta}$ represent all trainable parameters of the model. Our objective is to find the optimal parameter set $\widehat{\boldsymbol{\Theta}}$ that minimizes the following objective function $\mathcal{L}(\mathcal{D}, \boldsymbol{\Theta})$ for the given training data $\mathcal{D}$:

$$\mathcal{L}(\mathcal{D}, \boldsymbol{\Theta}) = -\frac{1}{|\mathcal{D}|} \sum_{(\boldsymbol{X},\boldsymbol{Y})\in\mathcal{D}} \log(p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})), \quad (1)$$

where $p(\boldsymbol{Y}|\boldsymbol{X}, \boldsymbol{\Theta})$ denotes the conditional probability of $\boldsymbol{Y}$ given $\boldsymbol{X}$.

In the standard supervised learning setting, the parallel data $\mathcal{D}$ comprise only "genuine" parallel data $\mathcal{D}_g$ (i.e., $\mathcal{D} = \mathcal{D}_g$). However, in GEC, incorporating pseudo data $\mathcal{D}_p$ that are generated from grammatical sentences $\boldsymbol{Y} \in \mathcal{T}$, where $\mathcal{T}$ represents *seed corpus* (i.e., a set of grammatical sentences), is common (Xie et al., 2018; Zhao et al., 2019; Grundkiewicz et al., 2019).

Our interest lies in the following three nontrivial aspects of Equation 1. **Aspect (i)**: multiple methods for generating pseudo data $\mathcal{D}_p$ are available (Section 3). **Aspect (ii)**: options for the seed corpus $\mathcal{T}$ are numerous. To the best of our knowledge, how the seed corpus domain affects the model performance is yet to be shown. We compare three corpora, namely, Wikipedia, Simple Wikipedia (SimpleWiki) and English Gigaword, as a first trial. Wikipedia and SimpleWiki have similar domains, but different grammatical complexities. Therefore, we can investigate how grammatical complexity affects model performance by comparing these two corpora. We assume that Gigaword contains the smallest amount of noise among the three corpora. We can therefore use Gigaword to investigate whether clean text improves model performance. **Aspect (iii)**: at least two major settings for incorporating $\mathcal{D}_p$ into the optimization of Equation 1 are available. One is to use the two datasets jointly by concatenating them as $\mathcal{D} = \mathcal{D}_g \cup \mathcal{D}_p$, which hereinafter we refer to as JOINT. The other is to use $\mathcal{D}_p$ for pretraining, namely, minimizing $\mathcal{L}(\mathcal{D}_p, \boldsymbol{\Theta})$ to acquire $\boldsymbol{\Theta}'$, and then fine-tuning the model by minimizing $\mathcal{L}(\mathcal{D}_g, \boldsymbol{\Theta}')$; hereinafter, we refer to this setting as PRETRAIN. We investigate these aspects through our extensive experiments (Section 4).

## 3 Methods for Generating Pseudo Data

In this section, we describe three methods for generating pseudo data. In Section 4, we experimentally compare these methods.

**BACKTRANS (NOISY) and BACKTRANS (SAMPLE)** Backtranslation for the EncDec model was proposed originally by Sennrich et al. (2016b). In backtranslation, a reverse model, which generates an ungrammatical sentence from a given grammatical sentence, is trained. The output of the reverse model is paired with the input and then used as pseudo data.

BACKTRANS (NOISY) is a variant of backtranslation that was proposed by Xie et al. (2018)[2]. This method adds $r\beta_{\text{random}}$ to the score of each hypothesis in the beam for every time step. Here, noise $r$ is sampled uniformly from the interval $[0, 1]$, and $\beta_{\text{random}} \in \mathbb{R}_{\geq 0}$ is a hyper-parameter that controls the noise scale. If we set $\beta_{\text{random}} = 0$, then BACKTRANS (NOISY) is identical to standard backtranslation.

BACKTRANS (SAMPLE) is another variant of backtranslation, which was proposed by Edunov et al. (2018) for MT. In BACKTRANS (SAMPLE), sentences are decoded by sampling from the distribution of the reverse model.

**DIRECTNOISE** Whereas BACKTRANS (NOISY) and BACKTRANS (SAMPLE) generate ungrammatical sentences with a reverse model, DIRECTNOISE injects noise "directly" into grammatical sentences (Edunov et al., 2018; Zhao et al., 2019). Specifically, for each token in the given sentence, this method probabilistically chooses one of the following operations: (i) masking with a placeholder token $\langle mask \rangle$, (ii) deletion, (iii) insertion of a random token, and (iv) keeping the original[3]. For each token, the choice is made based on the categorical distribution $(\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}})$.

## 4 Experiments

The goal of our experiments is to investigate **aspect (i)–(iii)** introduced in Section 2. To ensure that the experimental findings are applicable to GEC in general, we design our experiments by using the following two strategies: (i) we use an off-the-shelf EncDec model without any task-specific architecture or techniques; (ii) we conduct hyper-parameter tuning, evaluation and comparison of each method or setting on the validation set. At the end of experiments (Section 4.5), we summarize our findings and propose suitable settings. We then perform a single-shot evaluation of their performance on the test set.

[2] referred as "random noising" in Xie et al. (2018)

[3] The detailed algorithm is described in Appendix A.

| Dataset | #sent (pairs) | #refs. | Split | Scorer |
|---|---|---|---|---|
| BEA-train | 561,410 | 1 | train | - |
| BEA-valid | 2,377 | 1 | valid | ERRANT |
| CoNLL-2014 | 1,312 | 2 | test | ERRANT & $M^2$ scorer |
| JFLEG | 1,951 | 4 | test | GLEU |
| BEA-test | 4,477 | 5 | test | ERRANT |
| SimpleWiki* | 1,369,460 | - | - | - |
| Wikipedia* | 145,883,941 | - | - | - |
| Gigaword* | 131,864,979 | - | - | - |

Table 1: Summary of datasets used in our experiments. Dataset marked with "*" is a seed corpus $\mathcal{T}$.

| Method | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|
| Baseline | 46.6 | 23.1 | 38.8 |
| BACKTRANS (SAMPLE) | 44.6 | 27.4 | 39.6 |
| BACKTRANS (NOISY) | 42.5 | **31.3** | 39.7 |
| DIRECTNOISE | **48.9** | 25.7 | **41.4** |

Table 2: Performance of models on BEA-valid: a value in **bold** indicates the best result within the column. The seed corpus $\mathcal{T}$ is SimpleWiki.

### 4.1 Experimental Configurations

**Dataset** The BEA-2019 workshop official dataset[4] is the origin of the training and validation data of our experiments. Hereinafter, we refer to the training data as BEA-train. We create validation data (BEA-valid) by randomly sampling sentence pairs from the official validation split[5].

As a seed corpus $\mathcal{T}$, we use SimpleWiki[6], Wikipedia[7] or Gigaword[8]. We apply the noizing methods described in Section 3 to each corpus and generate pseudo data $\mathcal{D}_p$. The characteristics of each dataset are summarized in Table 1.

**Evaluation** We report results on BEA-valid, the official test set of the BEA-2019 shared task (BEA-test), the CoNLL-2014 test set (CoNLL-2014) (Ng et al., 2014), and the JFLEG test set (JFLEG) (Napoles et al., 2017). All reported results (except ensemble) are the average of five distinct trials using five different random seeds. We report the scores measured by ERRANT (Bryant et al., 2017; Felice et al., 2016) for BEA-valid, BEA-test, and CoNLL-2014. As the reference sentences of BEA-test are publicly unavailable, we evaluate the model outputs on `CodaLab`[9] for BEA-test. We also report results measured by the $M^2$ scorer (Dahlmeier and Ng, 2012) on CoNLL-2014 to compare them with those of previous studies. We use the GLEU metric (Napoles et al., 2015, 2016) for JFLEG.

**Model** We adopt the *Transformer* EncDec model (Vaswani et al., 2017) using the `fairseq` toolkit (Ott et al., 2019) and use the "Transformer (big)" settings of Vaswani et al. (2017).

**Optimization** For the JOINT setting, we optimize the model with Adam (Kingma and Ba, 2015). For the PRETRAIN setting, we pretrain the model with Adam and then fine-tune it on BEA-train using Adafactor (Shazeer and Stern, 2018)[10].

### 4.2 Aspect (i): Pseudo Data Generation

We compare the effectiveness of the BACKTRANS (NOISY), BACKTRANS (SAMPLE), and DIRECTNOISE methods for generating pseudo data. In DIRECTNOISE, we set $(\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}}) = (0.5, 0.15, 0.15, 0.2)$[11]. We use $\beta_{\text{random}} = 6$ for BACKTRANS (NOISY)[12]. In addition, we use (i) the JOINT setting and (ii) all of SimpleWiki as the seed corpus $\mathcal{T}$ throughout this section.

The results are summarized in Table 2. BACKTRANS (NOISY) and BACKTRANS (SAMPLE) show competitive values of $F_{0.5}$. Given this result, we exclusively use BACKTRANS (NOISY) and discard BACKTRANS (SAMPLE) for the rest of the experiments. The advantage of BACKTRANS (NOISY) is that its effectiveness in GEC has already been demonstrated by Xie et al. (2018). In addition, in our preliminary experiment, BACKTRANS (NOISY) decoded ungrammatical sentence 1.2 times faster than BACKTRANS (SAMPLE) did. We also use DIRECTNOISE because it achieved the best value of $F_{0.5}$ among all the methods.

### 4.3 Aspect (ii): Seed Corpus $\mathcal{T}$

We investigate the effectiveness of the seed corpus $\mathcal{T}$ for generating pseudo data $\mathcal{D}_p$. The three corpora (Wikipedia, SimpleWiki and Gigaword) are compared in Table 3. We set $|\mathcal{D}_p| = 1.4\text{M}$. The difference in $F_{0.5}$ is small, which implies that the seed corpus $\mathcal{T}$ has only a minor effect on the model performance. Nevertheless, Gigaword consistently outperforms the other two corpora. In particular,

[4] Details of the dataset is in Appendix B.

[5] The detailed data preparation process is in Appendix C.

[6] https://simple.wikipedia.org

[7] We used 2019-02-25 dump file at https://dumps.wikimedia.org/other/cirrussearch/.

[8] We used the English Gigaword Fifth Edition (LDC Catalog No.: LDC2011T07).

[9] https://competitions.codalab.org/competitions/20228

[10] The detailed hyper-parameters are listed in Appendix D.

[11] These values are derived from preliminary experiments (Appendix E).

[12] $\beta_{\text{random}} = 6$ achieved the best $F_{0.5}$ in our preliminary experiments (Appendix F).

| Method | Seed Corpus $\mathcal{T}$ | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|---|
| Baseline | N/A | 46.6 | 23.1 | 38.8 |
| BACKTRANS (NOISY) | Wikipedia | 43.8 | 30.8 | 40.4 |
| BACKTRANS (NOISY) | SimpleWiki | 42.5 | 31.3 | 39.7 |
| BACKTRANS (NOISY) | Gigaword | 43.1 | 33.1 | 40.6 |
| DIRECTNOISE | Wikipedia | 48.3 | 25.5 | 41.0 |
| DIRECTNOISE | SimpleWiki | 48.9 | 25.7 | 41.4 |
| DIRECTNOISE | Gigaword | 48.3 | 26.9 | 41.7 |

Table 3: Performance on BEA-valid when changing the seed corpus $\mathcal{T}$ used for generating pseudo data ($|\mathcal{D}_p| = 1.4$M).

DIRECTNOISE with Gigaword achieves the best value of $F_{0.5}$ among all the configurations.

### 4.4 Aspect (iii): Optimization Setting

We compare the JOINT and PRETRAIN optimization settings. We are interested in how each setting performs when the scale of the pseudo data $\mathcal{D}_p$ compared with that of the genuine parallel data $\mathcal{D}_g$ is (i) approximately the same ($|\mathcal{D}_p| = 1.4$M) and (ii) substantially bigger ($|\mathcal{D}_p| = 14$M). Here, we use Wikipedia as the seed corpus $\mathcal{T}$ instead of SimpleWiki or Gigaword for two reasons. First, SimpleWiki is too small for the experiment **(b)** (see Table 1). Second, the fact that Gigaword is not freely available makes it difficult for other researchers to replicate our results.

**(a) Joint Training or Pretraining** Table 4 presents the results. The most notable result here is that PRETRAIN demonstrates the properties of *more pseudo data and better performance*, whereas JOINT does not. For example, in BACKTRANS (NOISY), increasing $|\mathcal{D}_p|$ (1.4M $\rightarrow$ 14M) improves $F_{0.5}$ on PRETRAIN (41.1 $\rightarrow$ 44.5). By contrast, $F_{0.5}$ does not improve on JOINT (40.4 $\rightarrow$ 40.3). An intuitive explanation for this case is that when pseudo data $\mathcal{D}_p$ are substantially more than genuine data $\mathcal{D}_g$, the teaching signal from $\mathcal{D}_p$ becomes dominant in JOINT. PRETRAIN alleviates this problem because the model is trained with only $\mathcal{D}_g$ during fine-tuning. We therefore suppose that PRETRAIN is crucial for utilizing extensive pseudo data.

**(b) Amount of Pseudo Data** We investigate how increasing the amount of pseudo data affects the PRETRAIN setting. We pretrain the model with different amounts of pseudo data {1.4M, 7M, 14M, 30M, 70M}. The results in Figure 1 show that BACKTRANS (NOISY) has superior sample efficiency to DIRECTNOISE. The best model (pretrained with 70M BACKTRANS (NOISY)) achieves $F_{0.5} = 45.9$.

| Optimization | Method | $|\mathcal{D}_p|$ | Prec. | Rec. | $F_{0.5}$ |
|---|---|---|---|---|---|
| N/A | Baseline | 0 | 46.6 | 23.1 | 38.8 |
| PRETRAIN | BACKTRANS (NOISY) | 1.4M | 49.6 | 24.3 | 41.1 |
| PRETRAIN | DIRECTNOISE | 1.4M | 48.4 | 21.2 | 38.5 |
| JOINT | BACKTRANS (NOISY) | 1.4M | 43.8 | 30.8 | 40.4 |
| JOINT | DIRECTNOISE | 1.4M | 48.3 | 25.5 | 41.0 |
| PRETRAIN | BACKTRANS (NOISY) | 14M | 50.6 | 30.1 | 44.5 |
| PRETRAIN | DIRECTNOISE | 14M | 49.8 | 25.8 | 42.0 |
| JOINT | BACKTRANS (NOISY) | 14M | 43.0 | 32.3 | 40.3 |
| JOINT | DIRECTNOISE | 14M | 48.7 | 23.5 | 40.1 |

Table 4: Performance of the model with different optimization settings on BEA-valid. The seed corpus $\mathcal{T}$ is Wikipedia.

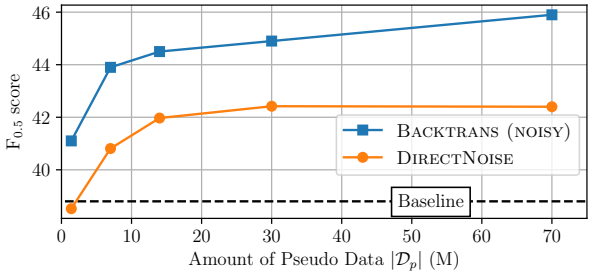


Figure 1: Performance on BEA-valid for different amounts of pseudo data ($|\mathcal{D}_p|$). The seed corpus $\mathcal{T}$ is Wikipedia.

### 4.5 Comparison with Current Top Models

The present experimental results show that the following configurations are effective for improving the model performance: (i) the combination of JOINT and Gigaword (Section 4.3), (ii) the amount of pseudo data $\mathcal{D}_p$ not being too large in JOINT (Section 4.4(a)), and (iii) PRETRAIN with BACKTRANS (NOISY) using large pseudo data $\mathcal{D}_p$ (Section 4.4(b)). We summarize these findings and attempt to combine PRETRAIN and JOINT. Specifically, we pretrain the model using 70M pseudo data of BACKTRANS (NOISY). We then fine-tune the model by combining BEA-train and relatively small DIRECTNOISE pseudo data generated from Gigaword (we set $|\mathcal{D}_p| = 250$K). However, the performance does not improve on BEA-valid. Therefore, the best approach available is simply to pretrain the model with large (70M) BACKTRANS (NOISY) pseudo data and then fine-tune using BEA-train, which hereinafter we refer to as PRETLARGE. We use Gigaword for the seed corpus $\mathcal{T}$ because it has the best performance in Table 3.

We evaluate the performance of PRETLARGE on test sets and compare the scores with the current top models. Table 5 shows a remarkable result, that is,

| Model | Ensemble | CoNLL-2014 ($M^2$ scorer) | | | CoNLL-2014 (ERRANT) | | | JFLEG | BEA-test (ERRANT) | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Prec. | Rec. | $F_{0.5}$ | Prec. | Rec. | $F_{0.5}$ | GLEU | Prec. | Rec. | $F_{0.5}$ |
| Chollampatt and Ng (2018) | | 60.9 | 23.7 | 46.4 | - | - | - | 51.3 | - | - | - |
| Junczys-Dowmunt et al. (2018) | | - | - | 53.0 | - | - | - | 57.9 | - | - | - |
| Grundkiewicz and Junczys-Dowmunt (2018) | | 66.8 | 34.5 | 56.3 | - | - | - | 61.5 | - | - | - |
| Lichtarge et al. (2019) | | 65.5 | 37.1 | 56.8 | - | - | - | 61.6 | - | - | - |
| Chollampatt and Ng (2018) | ✓ | 65.5 | 33.1 | 54.8 | - | - | - | 57.5 | - | - | - |
| Junczys-Dowmunt et al. (2018) | ✓ | 61.9 | 40.2 | 55.8 | - | - | - | 59.9 | - | - | - |
| Lichtarge et al. (2019) | ✓ | 66.7 | 43.9 | 60.4 | - | - | - | **63.3** | - | - | - |
| Zhao et al. (2019) | ✓ | 71.6 | 38.7 | 61.2 | - | - | - | 61.0 | - | - | - |
| Grundkiewicz et al. (2019) | ✓ | - | - | 64.2 | - | - | - | 61.2 | 72.3 | 60.1 | 69.5 |
| PRETLARGE | | 67.9 | 44.1 | 61.3 | 61.2 | 42.0 | 56.0 | 59.7 | 65.5 | 59.4 | 64.2 |
| PRETLARGE+SSE+R2L | ✓ | 72.4 | **46.1** | **65.0** | 67.3 | **44.0** | **60.9** | 61.4 | 72.1 | **61.8** | 69.8 |
| PRETLARGE+SSE+R2L+SED | ✓ | **73.3** | 44.2 | 64.7 | **68.1** | 42.1 | 60.6 | 61.2 | **74.7** | 56.7 | **70.2** |

Table 5: Comparison of our best model and current top models: a **bold** value indicates the best result within the column.

our PRETLARGE achieves $F_{0.5} = 61.3$ on CoNLL-2014. This result outperforms not only all previous single-model results but also all ensemble results except for that by Grundkiewicz et al. (2019).

To further improve the performance, we incorporate the following techniques that are widely used in shared tasks such as BEA-2019 and WMT[13]:

**Synthetic Spelling Error** (SSE) Lichtarge et al. (2019) proposed the method of probabilistically injecting character-level noise into the source sentence of pseudo data $\mathcal{D}_p$. Specifically, one of the following operations is applied randomly at a rate of 0.003 per character: deletion, insertion, replacement, or transposition of adjacent characters.

**Right-to-left Re-ranking** (R2L) Following Sennrich et al. (2016a, 2017); Grundkiewicz et al. (2019), we train four right-to-left models. The ensemble of four left-to-right models generate $n$-best candidates and their corresponding scores (i.e., conditional probabilities). We then pass each candidate to the ensemble of the four right-to-left models and compute the score. Finally, we re-rank the $n$-best candidates based on the sum of the two scores.

**Sentence-level Error Detection** (SED) SED classifies whether a given sentence contains a grammatical error. Asano et al. (2019) proposed incorporating SED into the evaluation pipeline and reported improved precision. Here, the GEC model is applied only if SED detects a grammatical error in the given source sentence. The motivation is that SED could potentially reduce the number of false-positive errors of the GEC model. We use the re-implementation of the BERT-based SED model (Asano et al., 2019).

Table 5 presents the results of applying SSE, R2L, and SED. It is noteworthy that PRETLARGE+SSE+R2L achieves state-of-the-art performance on both CoNLL-2014 ($F_{0.5} = 65.0$) and BEA-test ($F_{0.5} = 69.8$), which are better than those of the best system of the BEA-2019 shared task (Grundkiewicz et al., 2019). In addition, PRETLARGE+SSE+R2L+SED can further improve the performance on BEA-test ($F_{0.5} = 70.2$). However, unfortunately, incorporating SED decreased the performance on CoNLL-2014 and JFLEG. This fact implies that SED is sensitive to the domain of the test set since the SED model is fine-tuned with the official validation split of BEA dataset. We leave this sensitivity issue as our future work.

## 5 Conclusion

In this study, we investigated several aspects of incorporating pseudo data for GEC. Through extensive experiments, we found the following to be effective: (i) utilizing Gigaword as the seed corpus, and (ii) pretraining the model with BACKTRANS (NOISY) data. Based on these findings, we proposed suitable settings for GEC. We demonstrated the effectiveness of our proposal by achieving state-of-the-art performance on the CoNLL-2014 test set and the BEA-2019 test set.

## Acknowledgements

We thank the three anonymous reviewers for their insightful comments. We are deeply grateful to Takumi Ito and Tatsuki Kuribayashi for kindly sharing the re-implementation of BACKTRANS (NOISY). The work of Jun Suzuki was supported in part by JSPS KAKENHI Grant Number JP19104418 and AIRPF Grant Number 30AI036-8.

[13] http://www.statmt.org/wmt19/

## References


Hiroki Asano, Masato Mita, Tomoya Mizumoto, and Jun Suzuki. 2019. The AIP-Tohoku System at the BEA-2019 Shared Task. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 176–182.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural Machine Translation by Jointly Learning to Align and Translate. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

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 (ACL 2017)*, pages 793–805.

Shamil Chollampatt and Hwee Tou Ng. 2018. A Multilayer Convolutional Encoder-Decoder Neural Network for Grammatical Error Correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence (AAAI 2018)*, pages 5755–5762.

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 (NAACL 2012)*, pages 568–572.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a Large Annotated Corpus of Learner English: The NUS Corpus of Learner English. In *Proceedings of the 8th Workshop on Building Educational Applications Using NLP (BEA 2013)*, pages 22–31.

Sergey Edunov, Myle Ott, Michael Auli, and David Grangier. 2018. Understanding Back-Translation at Scale. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP 2018)*, pages 489–500.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic Extraction of Learner Errors in ESL Sentences Using Linguistically Enhanced Alignments. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Fluency Boost Learning and Inference for Neural Grammatical Error Correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (ACL 2018)*, pages 1055–1065.

Sylviane Granger. 1998. The computer learner corpus: A versatile new source of data for SLA research. In Sylviane Granger, editor, *Learner English on Computer*, pages 3–18. Addison Wesley Longman, London and New York.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near Human-Level Performance in Grammatical Error Correction with Hybrid Machine Translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 284–290.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2019)*, pages 252–263.

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A Nested Attention Neural Hybrid Model for Grammatical Error Correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (ACL 2017)*, pages 753–762.

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 (NAACL 2018)*, pages 595–606.

Diederik Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *Proceedings of the 3rd International Conference on Learning Representations (ICLR 2015)*.

Philipp Koehn and Rebecca Knowles. 2017. Six Challenges for Neural Machine Translation. In *Proceedings of the First Workshop on Neural Machine Translation*, pages 28–39.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora Generation for Grammatical Error Correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2019)*.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining Revision Log of Language Learning SNS for Automated Japanese Error Correction of Second Language Learners. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP 2011)*, pages 147–155.

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 (ACL & IJCNLP 2015)*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU Without Tuning. *arXiv preprint arXiv:1605.02592*.

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 (EACL 2017)*, pages 229–234.

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.

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 (NAACL 2019)*.

Rico Sennrich, Alexandra Birch, Anna Currey, Ulrich Germann, Barry Haddow, Kenneth Heafield, Antonio Valerio Miceli Barone, and Philip Williams. 2017. The university of Edinburgh's neural MT systems for WMT17. In *Proceedings of the Second Conference on Machine Translation (WMT 2017)*, pages 389–399.

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016b. Improving Neural Machine Translation Models with Monolingual Data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 86–96.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (ACL 2016)*, pages 1715–1725.

Noam Shazeer and Mitchell Stern. 2018. Adafactor: Adaptive Learning Rates with Sublinear Memory Cost. In *Proceedings of the 35th International Conference on Machine Learning (ICML 2018)*, pages 4603–4611.

Ilya Sutskever, Oriol Vinyals, and Quoc V. Le. 2014. Sequence to Sequence Learning with Neural Networks. In *Advances in Neural Information Processing Systems 28 (NIPS 2014)*, pages 3104–3112.

Christian Szegedy, Vincent Vanhoucke, Sergey Ioffe, Jon Shlens, and Zbigniew Wojna. 2016. Rethinking the inception architecture for computer vision. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR 2016)*, pages 2818–2826.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and Aspect Error Correction for ESL Learners Using Global Context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (ACL 2012)*, pages 198–202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention Is All You Need. In *Advances in Neural Information Processing Systems 31 (NIPS 2017)*, pages 5998–6008.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics (NAACL 2018)*, pages 619–628.

Helen Yannakoudakis, Øistein E. Andersen, Ardeshir Geranpayeh, Ted Briscoe, and Diane Nicholls. 2018. Developing an Automated Writing Placement system for ESL Learners. *Applied Measurement in Education*, 31(3):251–267.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics (ACL 2011)*, pages 180–189.

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 (NAACL 2019)*.

## A DIRECTNOISE Algorithm

The DIRECTNOISE algorithm is described in Algorithm 1 Here, $\boldsymbol{X}$ consists of sequence of $I$ tokens, namely, $\boldsymbol{X} = (x_1, \ldots, x_I)$ where $x_i$ denotes $i$-th token of $\boldsymbol{X}$. Similarly, $\boldsymbol{Y}$ consists of sequence of $J$ tokens, namely, $\boldsymbol{Y} = (y_1, \ldots, y_J)$ where $y_j$ denotes $j$-th token of $\boldsymbol{Y}$.

**Algorithm 1:** DIRECTNOISE Algorithm

**Data:** Grammatical sentence $\boldsymbol{Y} \in \mathcal{T}$
**Result:** Pseudo Corpus $\mathcal{D}_p$

$\mathcal{D}_p = \{\}$ // create empty set
$\boldsymbol{\mu} = \{\mu_{\text{mask}}, \mu_{\text{deletion}}, \mu_{\text{insertion}}, \mu_{\text{keep}}\}$ s.t. $\Sigma\boldsymbol{\mu}=1$
**for** $\boldsymbol{Y} \in \mathcal{T}$ **do**
  $\boldsymbol{X} = (\,)$
  **for** $j \in (1, \ldots, J)$ **do**
    $action \sim Cat(action|\boldsymbol{\mu})$
    **if** *action is keep* **then**
      append $y_j$ to $\boldsymbol{X}$
    **else if** *action is mask* **then**
      append $\langle mask \rangle$ to $\boldsymbol{X}$
    **else if** *action is deletion* **then**
      continue
    **else if** *action is insertion* **then**
      append $y_j$ to $\boldsymbol{X}$
      $w = sample\_from\_unigram\_distribution(\mathcal{D}_g)$
      append $w$ to $\boldsymbol{X}$
  $\mathcal{D}_p = \mathcal{D}_p \cup \{(\boldsymbol{X}, \boldsymbol{Y})\}$

## B BEA-2019 Workshop Official Dataset

The BEA-2019 Workshop official dataset consists of following corpora: the First Certificate in English corpus (Yannakoudakis et al., 2011), Lang-8 Corpus of Learner English (Lang-8) (Mizumoto et al., 2011; Tajiri et al., 2012), the National University of Singapore Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), and W&I+LOCNESS (Yannakoudakis et al., 2018; Granger, 1998). The data is publicly available at https://www.cl.cam.ac.uk/research/nl/bea2019st/.

## C Data Preparation Process

The training data (BEA-train) is tokenized using spaCy tokenizer[14]. We used en_core_web_sm-2.1.0 model[15]. We remove sentence pairs that have identical source and target sentences from the training set, following (Chollampatt and Ng, 2018). Then we acquire subwords from target sentence through byte-pair-encoding (BPE) (Sennrich et al., 2016c) algorithm. We used subword-nmt implementation[16]. We apply BPE splitting to both source and target text. The number of merge operation is set to 8,000.

[14] https://spacy.io/
[15] https://github.com/explosion/spacy-models/releases/tag/en_core_web_sm-2.1.0
[16] https://github.com/rsennrich/subword-nmt

## D Hyper-parameter Settings

| Configurations | Values |
|---|---|
| Model Architecture | Transformer (Vaswani et al., 2017) ("big" setting) |
| Optimizer | Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$) |
| Learning Rate Schedule | Same as described in Section 5.3 of Vaswani et al. (2017) |
| Number of Epochs | 40 |
| Dropout | 0.3 |
| Stopping Criterion | Train model for 40 epochs. During the training, save model parameter for every 500 updates. Then take average of last 20 checkpoints. |
| Gradient Clipping | 1.0 |
| Loss Function | Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016) |
| Beam Search | Beam size 5 with length-normalization |

Table 6: Hyper-parameter for JOINT optimization

| Configurations | Values |
|---|---|
| **Pretraining** | |
| Model Architecture | Transformer (Vaswani et al., 2017) ("big" setting) |
| Optimizer | Adam (Kingma and Ba, 2015) ($\beta_1 = 0.9, \beta_2 = 0.98, \epsilon = 1 \times 10^{-8}$) |
| Learning Rate Schedule | Same as described in Section 5.3 of Vaswani et al. (2017) |
| Number of Epochs | 10 |
| Dropout | 0.3 |
| Gradient Clipping | 1.0 |
| Loss Function | Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016) |
| **Fine-tuning** | |
| Model Architecture | Transformer (Vaswani et al., 2017) ("big" setting) |
| Optimizer | Adafactor (Shazeer and Stern, 2018) |
| Learning Rate Schedule | Constant learning rate of $3 \times 10^{-5}$ |
| Number of Epochs | 30 |
| Dropout | 0.3 |
| Stopping Criterion | Use the model with the best validation perplexity on BEA-valid |
| Gradient Clipping | 1.0 |
| Loss Function | Label smoothed cross entropy (smoothing value: $\epsilon_{ls} = 0.1$) (Szegedy et al., 2016) |
| Beam Search | Beam size 5 with length-normalization |

Table 7: Hyper-parameter for PRETRAIN optimization

## E Mask Probability of DIRECTNOISE

In this paper, we exclusively focused on the effectiveness of $\mu_{mask}$, and therefore we deliberately fixed $\mu_{\text{keep}} = 0.2$, and used $\mu_{\text{insertion}} = \mu_{\text{deletion}} = (1 - \mu_{\text{keep}} - \mu_{\text{mask}})/2$

We investigated the effectiveness of changing mask probability $\mu_{\text{mask}}$ of BACKTRANS (NOISY) by evaluating the model performance on BEA-valid. We used entire SimpleWiki as the seed corpus $\mathcal{T}$. The result is summarized in Figure 2. Here, increasing $\mu_{\text{mask}}$ within the range of $0.1 < \mu_{\text{mask}} < 0.5$ slightly improved the performance. Thus, used $\mu_{\text{mask}} = 0.5$ in the experiment (Section 4).

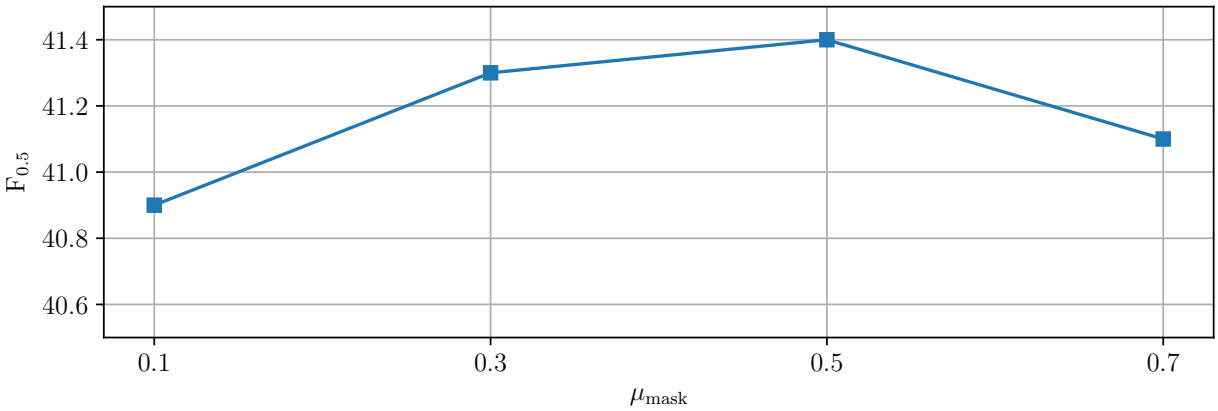


Figure 2: Performance of the model on BEA-valid as parameter of DIRECTNOISE ($\mu_{\text{mask}}$) is varied.

## F Noise Strength of BACKTRANS (NOISY)

We investigated the effectiveness of varying $\beta_{\text{random}}$ hyper-parameter of BACKTRANS (NOISY) by evaluating its performance on BEA-valid (Figure 3). We used entire SimpleWiki as the seed corpus $\mathcal{T}$. The figure shows that the performance of backtranslation without noise ($\beta_{\text{random}} = 0$) is worse than the baseline. We believe that when there is no noise, reverse-model becomes too conservative to generate grammatical error, as discussed by Xie et al. (2018). Thus, the generated pseudo data cannot provide useful teaching signal for the model.

In terms of the scale of the noise, $\beta_{\text{random}} = 6$ is the best value for BACKTRANS (NOISY). Thus, we used this value in the experiment (Section 4).

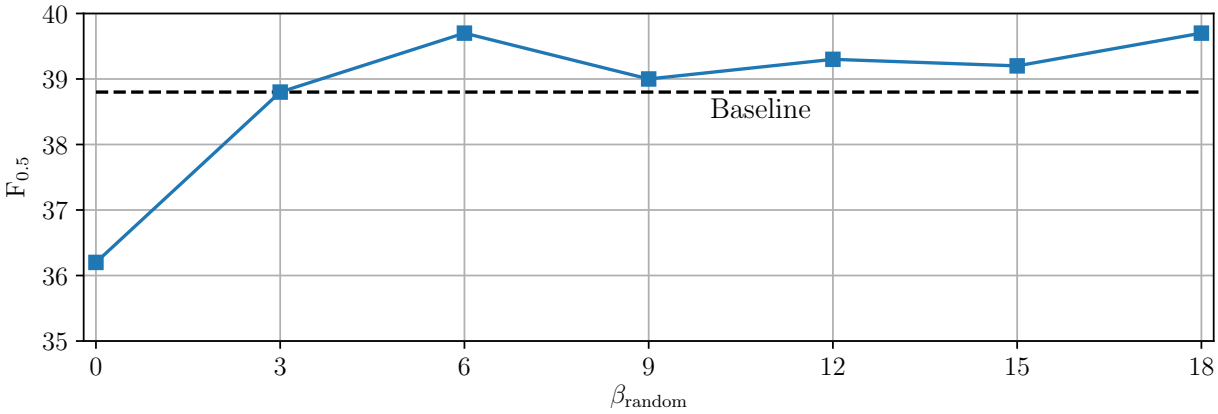


Figure 3: Performance of the model on BEA-valid as parameter of BACKTRANS (NOISY) ($\beta_{\text{random}}$) is varied.

## G Examples of Noisy Sentences

Figure 4 shows examples of noisy sentences that are generated by BACKTRANS (NOISY) and DIRECTNOISE.

| | |
|---|---|
| Original: | He died there , but the death date is not clear . |
| BACKTRANS (NOISY): | He died at there , but death date is not clear . |
| DIRECTNOISE: | ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ , 2 but ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ is not ⟨*mask*⟩ ⟨*mask*⟩ |
| Original: | On seeing her his joy knew no bounds . |
| BACKTRANS (NOISY): | On seeing her joyful knew no bounds . |
| DIRECTNOISE: | ⟨*mask*⟩ ⟨*mask*⟩ her crahis ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ bke ⟨*mask*⟩ . |
| Original: | Gre@@ en@@ space Information for G@@ rea@@ ter London . |
| BACKTRANS (NOISY): | The information for Gre@@ en@@ space information about G@@ rea@@ ter London . |
| DIRECTNOISE: | ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ for ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ |
| Original: | The cli@@ p is mixed with images of Toronto streets during power failure . |
| BACKTRANS (NOISY): | The cli@@ p is mix with images of Toronto streets during power failure . |
| DIRECTNOISE: | The ⟨*mask*⟩ is mixed ⟨*mask*⟩ images si@@ of The ⟨*mask*⟩ streets large ⟨*mask*⟩ power R@@ failure place ⟨*mask*⟩ |
| Original: | At the in@@ stitute , she introduced tis@@ sue culture methods that she had learned in the U.@@ S. |
| BACKTRANS (NOISY): | At in@@ stitute , She introduced tis@@ sue culture method that she learned in U.@@ S. |
| DIRECTNOISE: | ⟨*mask*⟩ the the ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ tis@@ culture R@@ methods , she P ⟨*mask*⟩ the s U.@@ ⟨*mask*⟩ |

Figure 4: Examples of sentences generated by BACKTRANS (NOISY) and DIRECTNOISE methods.

Figure 5 shows examples generated by DIRECTNOISE, when changing the mask probability ($\mu_{\text{mask}}$).

| $\mu_{\text{mask}}$ | Output Sentence |
|---|---|
| N/A | He threw the sand@@ wi@@ ch at his wife . |
| 0.1 | He ale threw , ch his ne@@ wife dar@@ ⟨*mask*⟩ |
| 0.3 | ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ch at ament his Research . |
| 0.5 | He o threw the sand@@ ch ⟨*mask*⟩ his ⟨*mask*⟩ . |
| 0.7 | ⟨*mask*⟩ ⟨*mask*⟩ sand@@ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ ⟨*mask*⟩ wife ⟨*mask*⟩ |

Figure 5: Examples generated when varying $\mu_{\text{mask}}$. N/A denotes original text.

## H Performance of the Model without Fine-tuning

PRETRAIN setting undergoes two optimization steps, namely, pretraining with pseudo data $\mathcal{D}_p$ and fine-tuning with genuine parallel data $\mathcal{D}_g$. We report the performance of models with pretraining only (Figure 6).

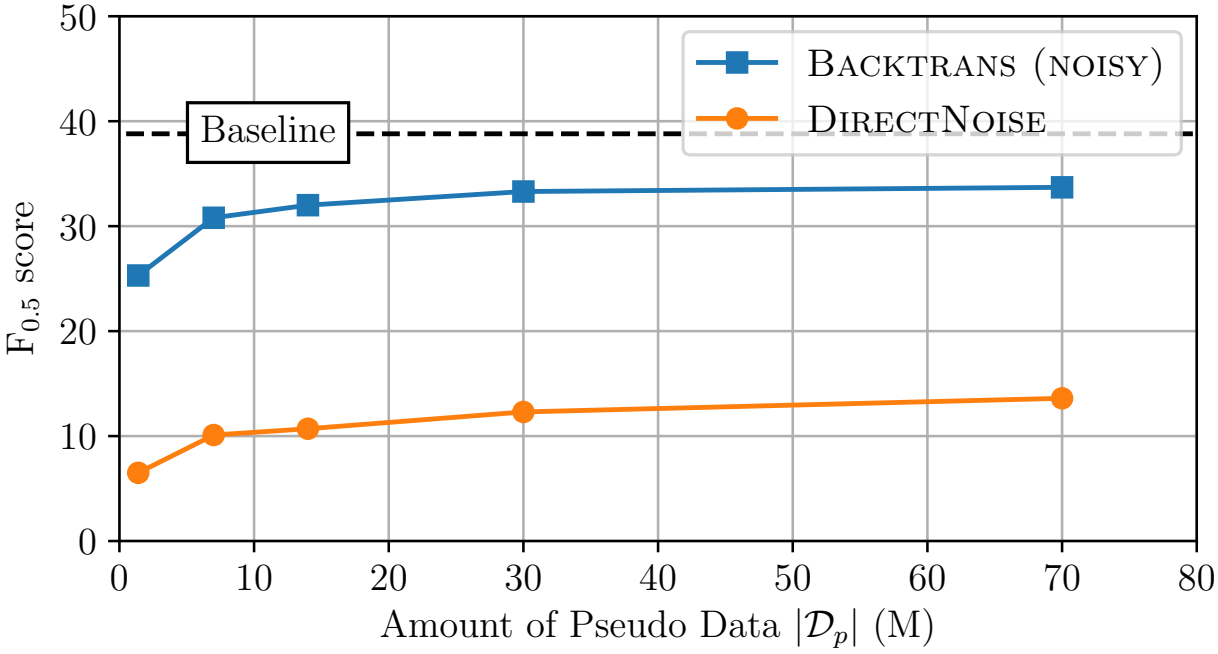


Figure 6: Performance on BEA-valid when varying the amount of pseudo data ($|\mathcal{D}_p|$)