# Noising and Denoising Natural Language: Diverse Backtranslation for Grammar Correction

**Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Y. Ng, Dan Jurafsky**
Computer Science Department, Stanford University
{zxie,genthial,stanxie,ang}@cs.stanford.edu,jurafsky@stanford.edu

## Abstract

Translation-based methods for grammar correction that directly map noisy, ungrammatical text to their clean counterparts are able to correct a broad range of errors; however, such techniques are bottlenecked by the need for a large parallel corpus of noisy and clean sentence pairs. In this paper, we consider synthesizing parallel data by noising a clean monolingual corpus. While most previous approaches introduce perturbations using features computed from local context windows, we instead develop error generation processes using a neural sequence transduction model trained to translate clean examples to their noisy counterparts. Given a corpus of clean examples, we propose beam search noising procedures to synthesize additional noisy examples that human evaluators were nearly unable to discriminate from nonsynthesized examples. Surprisingly, when trained on additional data synthesized using our best-performing noising scheme, our model approaches the same performance as when trained on additional nonsynthesized data.

## 1 Introduction

Correcting noisy, ungrammatical text remains a challenging task in natural language processing. Ideally, given some piece of writing, an error correction system would be able to fix minor typographical errors, as well as grammatical errors that involve longer dependencies such as nonidiomatic phrasing or errors in subject-verb agreement. Existing methods, however, are often only able to correct highly local errors, such as spelling errors or errors involving articles or prepositions.

Classifier-based approaches to error correction are limited in their ability to capture a broad range of error types (Ng et al., 2014). Machine translation-based approaches—that instead translate noisy, ungrammatical sentences to clean, corrected sentences—can flexibly handle a large variety of errors; however, such approaches are bottlenecked by the need for a large dataset of source-target sentence pairs.

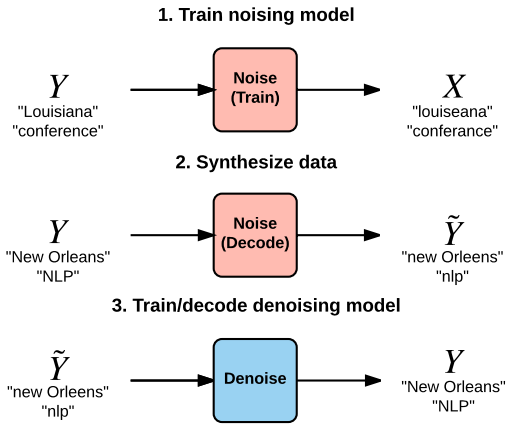


Figure 1: Overview of method. We first train a noise model on a seed corpus, then apply noise during decoding to synthesize data that is in turn used to train the denoising model.

To address this data sparsity problem, we propose methods for synthesizing noisy sentences from clean sentences, thus generating an additional artificial dataset of noisy and clean sentence pairs. A simple approach to noise clean text is to noise individual tokens or bigrams, for example by replacing each token with a random draw from the unigram distribution. This type of approach, however, tends to generate highly unrealistic noise and fails to capture phrase-level phenomena. Other rule-based approaches fail to capture a diverse set of error types.

We consider a method inspired by the backtranslation procedure for machine translation (Sennrich et al., 2015). Our method combines a neural sequence transduction trained on a seed corpus of clean→noisy pairs with *beam search*

*noising* procedures to produce more diversity in the decoded outputs. This technique addresses two issues with existing synthesis techniques for grammar correction:

1. By using a neural model trained end-to-end on a large corpus of noisy and clean sentences, the model is able to generate rich, diverse errors that better capture the noise distribution of real data.

2. By encouraging diversity through applying noise to hypotheses during decoding, we avoid what we refer to as the *one-to-many* problem, where decoding from a model trained on clean→noisy examples results in overly clean output, since clean subphrases still form the majority of noisy examples.

We perform experiments using several noising methods to validate these two claims, yielding gains on two benchmarks. Our main empirical result is that, starting with only clean news data and models trained on a parallel corpus of roughly 1.3 million sentences, we can train models with additional synthesized data that nearly match the performance of models trained on 3 million nonsynthesized examples.

## 2 Related work

**Noising** While for images, there are natural noising primitives such as rotations, small translational shifts, and additive Gaussian noise, similar primitives are not as well developed for text data. Similarly, while denoising autoencoders for images have been shown to help with representation learning (Vincent et al., 2010), similar methods for learning representations are not well developed for text. Some recent work has proposed noising—in the form of dropping or replacing individual tokens—as a regularizer when training sequence models, where it has been demonstrated to have a smoothing effect on the softmax output distribution (Bowman et al., 2015; Xie et al., 2017; Dai and Le, 2015; Kumar et al., 2015).

**Grammar correction** Recent work by Chollampatt and Ng (2018) has achieved impressive performance on the benchmarks we consider using convolutional encoder-decoder models. Previous work using data synthesis for grammatical error correction (GEC) has introduced errors by examining the distribution of error types, then applying errors according to those distributions together with lexical or part-of-speech features based on a small context window (Brockett et al., 2006; Felice, 2016). While these methods can introduce many possible edits, they are not as flexible as our approach inspired by the backtranslation procedure for machine translation (Sennrich et al., 2015). This is important as neural language models not explicitly trained to track long-range linguistic dependencies can fail to capture even simple noun-verb errors (Linzen et al., 2016). Recently, in the work perhaps most similar to ours, Rei et al. (2017) propose using statistical machine translation and backtranslation along with syntactic patterns for generating errors, albeit for the error detection task.

**Neural machine translation** Recent end-to-end neural network-based approaches to machine translation have demonstrated strong empirical results (Sutskever et al., 2014; Cho et al., 2014). Building off of these strong results on machine translation, we use neural encoder-decoder models with attention (Bahdanau et al., 2014) for both our data synthesis (noising) and grammar correction (denoising) models. Although many recent works on NMT have focused on improving the neural network architecture, the model architecture is orthogonal to the contributions in this work, where we instead focus on data synthesis. In parallel to our work, work on machine translation without parallel corpora has also explored applying noise to avoid copying when pretraining autoencoders by swapping adjacent words (Lample et al., 2017; Artetxe et al., 2017).

**Diverse decoding** Key to the data generation procedure we describe is adding noise to the scores of hypotheses during beam search–otherwise, decoded outputs tend to contain too few errors. This is inspired by work in dialogue, in which neural network models tend to produce common, overly generic responses such as "*I don't know*" (Sordoni et al., 2015; Serban et al., 2015). To mitigate this issue, Li et al. (2015) and others have proposed methods to increase the diversity of neural network outputs. We adopt a similar approach to Li et al. (2015) to generate noisier hypotheses during decoding.

## 3 Method

We first briefly describe the neural model we use, then detail the noising schemes we apply when synthesizing examples.

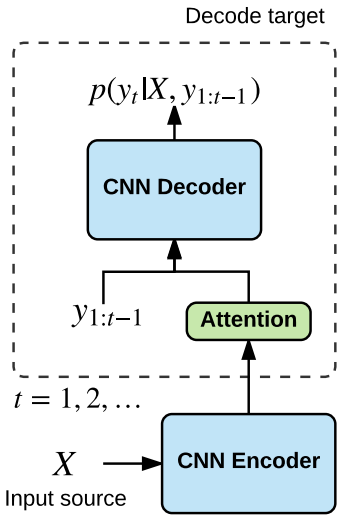


Figure 2: Model architecture used for both noising and denoising networks.

### 3.1 Model

In order to generate noisy examples as well as to translate ungrammatical examples to their corrected counterparts, we need to choose a sequence transduction model. Based off their strong empirical performance, we use a neural network-based model for this work.

Our method uses two neural encoder-decoder models:

1. The first is the *noising* model, which, given a clean sentence, is used to generate a noised version of that sentence. This model is trained on a *seed corpus* of parallel clean→noisy sentences.

2. The second is the *denoising* model, which, given a noisy, ungrammatical sentence, generates the clean, corrected sentence.

For both models, we use the same convolutional encoder-decoder to model

$$p(Y|X) = \prod_{t=1}^{T_Y} p(y_t|X, y_{1:t-1}; \theta)$$

where $X = (x_1, x_2, \ldots, x_{T_X})$ is the source sequence and $Y = (y_1, y_2, \ldots, y_{T_Y})$ the corresponding target sequence, and we minimize the training loss

$$\ell(\theta) = -\log \sum_{t=1}^{T_Y} p(y_t|X, y_{1:t-1}; \theta)$$

thus maximizing log-likelihood. The model architecture we use is similar to that described by Kalchbrenner et al. (2016) and Gehring et al. (2017). Gated convolutions are applied with masking—to avoid peeking at future inputs when training using teacher forcing—such that they form an autoregressive network similar to a recurrent neural network with gated hidden units. This architecture was selected so that training steps could be parallelized across the time dimension through the use of convolutions. However, we emphasize that the architecture is not a focus of this paper, and we would expect that RNN architectures with LSTM cells would achieve similar results. For simplicity and to avoid handling out-of-vocabulary words, we use character-level tokenization. Figure 2 illustrates the model architecture.

### 3.2 Noising

The amount of parallel data is often the limiting factor in the performance of neural network systems. In order to obtain more parallel examples for the grammar correction task, we take clean text $Y$ and apply noise, yielding noisy text $\tilde{Y}$, then train a denoising model to map from $\tilde{Y}$ back to $Y$. The noising process used to generate $\tilde{Y}$ greatly affects final performance. First, we consider noising methods which we use as our baselines, as well as the drawbacks for each method.

- **appending clean examples**: We first consider simply appending clean examples with no noise applied to both the source and the target. The aim is for the decoder to learn a better language model when trained on additional clean text, similar to the motivation described in Dai and Le (2015). However, for the models we consider, the attention mechanism allows copying of source to target. Thus the addition of examples where source and target are identical data may also cause the model to become too conservative with edits and thus reduce the recall of the system.

- **token noising**: Here we simply consider a context window of at most two characters or words and allow word/character deletions and transpositions.

  First, for every *character* in each word we sample deletions, followed by transpositions. Then we sample deletions and transpositions for every *word* in the sentence. Deletion and transposition probabilities were selected such

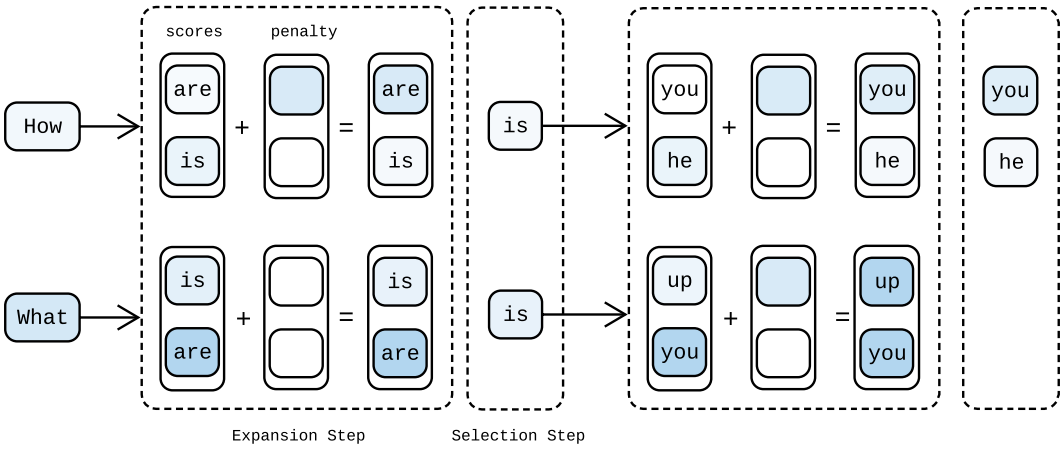


Figure 3: Illustration of random noising with beam width 2. Darker shading indicates less probable expansions. In this example, greedy decoding would yield "How are you". Applying noise penalties, however, results in the hypotheses "How is you/he". Note that applying a penalty does not always result in an expansion falling off the beam.

that overall character and word-level edit distances roughly matched the edit distances between clean and noisy examples in our parallel seed corpus. While this method is fast to apply, it tends to produce highly unrealistic errors leading to a mismatch between the synthesized and real parallel data.

- **reverse noising**: For reverse noising, we simply train a reverse model from $Y \rightarrow X$ using our parallel noisy-clean corpus and run standard beam search to generate noisy targets $\tilde{Y}$ from clean inputs $Y$. However, we find vanilla reverse noising tends to be too conservative. This is due to the *one-to-many* problem where a clean sentence has many possible noisy outputs which mostly consist of clean phrases. The output then contains far fewer errors on average than the original noisy text.

To address the drawback of the reverse noising scheme, we draw inspiration from ideas for increasing diversity of outputs in dialogue (Li et al., 2016). During the beam search procedure, we add noise to the scores of hypotheses on the beam to encourage decoding to stray from the greedy output. Recall that during beam search, we iteratively grow a set of hypotheses $\mathcal{H} = \{h_1, h_2, \ldots\}$, only keeping the top hypotheses after each step of decoding according to some scoring function $s(h)$. Extending the reverse noising scheme, the *beam search noising* schemes we consider are:

- **rank penalty noising** We directly apply the method of Li et al. (2016). At every step of the search procedure, siblings from the same parent are penalized by adding $k\beta_{\text{rank}}$ to their scores, where $k$ is their rank (in descending log-likelihood) amongst their siblings and $\beta_{\text{rank}}$ is a penalty hyperparameter corresponding to some log-probability.

- **top penalty noising** Only the top (most-probable) hypothesis $h_{\text{top}}$ of the beam is penalized by adding $\beta_{\text{top}}$ to its score $s(h_{\text{top}})$.

- **random noising** Every hypothesis is penalized by adding $r\beta_{\text{random}}$ to its score, where $r$ is drawn uniformly from the interval $[0, 1]$. For sufficiently large $\beta_{\text{random}}$, this leads to a random shuffling of the ranks of the hypotheses according to their scores.

An illustration of the random noising algorithm is shown in Figure 3. Note that although rank penalty noising should encourage hypotheses whose parents have similar scores to remain on the beam, it can also tend to leave the hypothesis from greedy decoding on the beam in the case where softmax output distributions are highly peaked. This is much more of an issue for tasks that involve significant copying of source to target, such as grammar correction. Note also that the random noising can yield more diverse outputs than top penalty noising, depending on the probability with which each is applied. All of the beam search noising methods described are intended to increase the diversity and the amount of noise in the synthe-

| Corpus | Sent. Pairs |
|---|---|
| CoNLL 2014 | 60K |
| Lang-8 | 1.3M |
| Lang-8 expanded | 3.3M |
| synthesized (NYT 2007) | 1.0M |
| base (CoNLL + L8) | 1.3M |
| expanded (CoNLL + L8 expanded) | 3.3M |

Table 1: Summary of training corpora.

sized outputs $\tilde{Y}$. By performing beam search noising, we can produce errors such as those shown in Table 4.

### 3.3 Denoising

Once noised data has been generated, *denoising* simply involves using a neural sequence transduction model to backtranslate the noised text to the original clean text. For denoising, during decoding we apply length normalization as well as a coverage penalty to the scoring function $s(h)$ (Wu et al., 2016). The final scoring function also incorporates a 5-gram language model trained on a subset of Common Crawl, estimated with Kneser-Ney smoothing using KenLM (Heafield, 2011). We incorporate the language model during final reranking by modifying the score for a completed hypothesis $s(h)$ to be

$$s_{\text{LM}}(h) = s(h) + \lambda \log p_{\text{LM}}(h)$$

where $\lambda$ is a hyperparameter and $p_{\text{LM}}(h)$ is given by the language model.

## 4 Experiments

To determine the effectiveness of the described noising schemes, we synthesize additional data using each and evaluate the performance of models trainined using the additional data on two benchmarks.

**Datasets** For training our sequence transduction models, we combine the publicly available English Lang-8 dataset, a parallel corpus collected from a language learner forum, with training data from the CoNLL 2014 challenge (Mizumoto et al., 2011; Ng et al., 2014). We refer to this as the "base" dataset. Junczys-Dowmunt and Grundkiewicz (2016) additionally scraped 3.3M pairs of sentences from Lang-8. Although this expanded dataset, which we call the "expanded" dataset, is not typically used when comparing performance on grammar correction benchmarks, we use it instead to compare performance when training on additional synthesized data versus nonsynthesized data. For clean text to be noised, we use the LDC New York Times corpus for 2007, which yields roughly 1 million sentences. A summary of the data used for training is given in Table 1.

We use the CoNLL 2013 evaluation set as our development set in all cases (Ng et al., 2013). Our test sets are the CoNLL 2014 evaluation set and the JFLEG test set (Ng et al., 2014; Napoles et al., 2017). Because CoNLL 2013 only has a single set of gold annotations while CoNLL 2014 has two, performance metrics tend to be significantly higher on CoNLL 2014. We report precision, recall, and $F_{0.5}$ score, which is standard for the task, as precision is valued over recall. On JFLEG, we report results with the GLEU metric (similar to BLEU) developed for the dataset.

**Training and decoding details** All models are trained using stochastic gradient descent with annealing based on validation perplexity on a small held-out subset of the Lang-8 corpus. We apply both dropout and weight decay regularization. We observed that performance tended to saturate after 30 epochs. Decoding is done with a beam size of 8; in early experiments, we did not observe significant gains with larger beam sizes (Koehn and Knowles, 2017).

### 4.1 CoNLL

Results for the CoNLL 2013 (dev) and 2014 (test) datasets but with and without language model reranking are given in Table 2. In general, adding noised data helps, while simply adding clean data leads the model to be too conservative. Overall, we find that the random noising scheme yields the most significant gain of $4.5$ $F$-score. Surprisingly, we find that augmenting the base dataset with synthesized data generated with random noising yields nearly the same performance when compared to using only nonsynthesized examples. To determine whether this might be due to overfitting, we reduced the dropout rate when training on the "expanded" dataset, but did not observe better results.

The random noising scheme achieves the best performance, while the top noising scheme matches the best performance on the development set but not the test set. We believe this is due to a mismatch between the CoNLL 2013 dev and 2014

| Method | Dev (no LM) | | | Dev | | | Test | | |
|---|---|---|---|---|---|---|---|---|---|
| | $P$ | $R$ | $F_{0.5}$ | $P$ | $R$ | $F_{0.5}$ | $P$ | $R$ | $F_{0.5}$ |
| none | 50.7 | 10.5 | 28.7 | 48.4 | 17.2 | 35.5 | 52.7 | 27.5 | 44.5 |
| clean | 56.1 | 9.4 | 28.1 | 47.5 | 16.9 | 34.8 | 52.3 | 27.5 | 44.3 |
| token | 49.7 | 11.9 | 30.4 | 47.7 | 18.7 | 36.4 | 51.4 | 30.3 | 45.1 |
| reverse | 53.1 | 13.0 | 32.8 | 50.5 | 19.1 | 38.0 | 54.7 | 29.6 | 46.8 |
| rank | 51.3 | 12.3 | 31.4 | 51.0 | 18.3 | 37.6 | 54.3 | 29.3 | 46.4 |
| top | 49.1 | 17.4 | 36.0 | 47.7 | 23.9 | **39.8** | 50.9 | 34.7 | 46.6 |
| random | 50.0 | 17.9 | **36.8** | 48.9 | 23.0 | **39.9** | 54.2 | 35.4 | **49.0** |
| expanded | 64.4 | 11.2 | 33.0 | 54.9 | 20.0 | 40.7 | 57.2 | 32.0 | **49.4** |
| Yuan and Briscoe (2016) | — | — | — | — | — | — | — | — | 39.9 |
| Ji et al. (2017) | — | — | 28.6 | — | — | 33.5 | — | — | 45.2 |
| Junczys-Dowmunt et al. (2016) | — | — | — | — | — | — | 61.3 | 28.0 | 49.5 |
| Chollampatt and Ng (2018) | — | — | — | — | — | — | 65.5 | 33.1 | **54.8** |

Table 2: Results on CoNLL 2013 (Dev) and CoNLL 2014 (Test) sets. All results use the "base" parallel corpus of 1.3M sentence pairs along with additional synthesized data (totaling 2.3M sentence pairs) except for "expanded", which uses 3.3M nonsynthesized sentence pairs (and no synthesized data).

tets sets. Since the 2013 dev set has only a single annotator, methods are encouraged to target higher recall, such that the top noising scheme was optimized for precision over recall. To check this, we ran decoding on CoNLL 2014 using the best dev settings with no language model, and found that the top noising scheme yielded an $F_{0.5}$-score of 45.2, behind only random (47.1) and ahead of token (42.0) and reverse (43.9) noising. Overall, we find the data synthesis method we describe to yield large gains in recall.

For completeness, we also compare to other state-of-the-art systems, such as the phrase-based machine translation system by Junczys-Dowmunt and Grundkiewicz (2016), who performed parameter tuning with sparse and dense features by cross-validation on the CoNLL 2014 training set. Chollampatt and Ng (2018) achieve even higher state-of-the-art results using the neural machine translation model of Gehring et al. (2017) along with improvements to the reranking procedure.

### 4.2 JFLEG

Recently, Napoles et al. (2017) introduced the JFLEG dataset, intended to evaluate the fluency of grammar correction systems rather than simply the precision and recall of edits. The evaluation metric proposed is GLEU, a variant of BLEU score. Most results for this task were reported with hyperparameter settings from the CoNLL task; hence we report results with the best settings on our CoNLL 2013 dev set. Results are shown in Table 3[1]. Token noising performs surprisingly well; we suspect this is because a significant portion of errors in the JFLEG dataset are spelling errors, as demonstrated from strong gains in performance by using a spelling checker reported by Chollampatt and Ng (2018).

## 5 Discussion

Our experiments illustrate that synthesized parallel data can yield large gains on the grammar correction task. However, what factors make for an effective data synthesis technique? We consider the properties of the noising scheme and the corresponding data that lead to better performance.

### 5.1 Realism and Human Evaluation

First, we manually compare each of the different noising methods to evaluate how "realistic" the errors introduced are. This is reminiscent of the generative adversarial network setting (Goodfellow et al., 2014), where the generator seeks to produce samples that fool the discriminator. Here the discriminator is a human evaluator who, given the clean sentence $Y$, tries to determine which of two sentences $X$ and $\tilde{Y}$ is the true noisy sentence, and which is the synthesized sentence. To be clear,

[1]Comparisons taken from https://github.com/keisks/jfleg

| Scheme | $P$ | $R$ | $F_{0.5}$ | GLEU |
|---|---|---|---|---|
| none | 68.9 | 44.2 | 62.0 | 53.9 |
| clean | 69.2 | 42.8 | 61.6 | 54.1 |
| token | 69.2 | 47.6 | **63.5** | 55.9 |
| reverse | 69.1 | 42.1 | 61.3 | 53.8 |
| rank | 68.3 | 43.3 | 61.2 | 54.4 |
| top | 67.3 | 48.2 | 62.4 | 55.5 |
| random | 69.1 | 48.5 | **63.7** | **56.6** |
| expanded | 72.7 | 45.9 | **65.1** | **56.2** |
| Sakaguchi et al. (2017)† | | | | 54.0 |
| Ji et al. (2017) | | | | 53.4 |
| Yuan and Briscoe (2016) | | | | 52.1 |
| Junczys-Dowmunt et al. (2016) | | | | 51.5 |
| Chollampatt and Ng (2018) | | | | **57.5** |

Table 3: Results on the JFLEG test set (we use best hyperparameter settings from CoNLL dev set). GLEU is a variant of BLEU developed for this task; higher is better (Napoles et al., 2017). †Tuned to JFLEG dev set.

we do not train with a discriminator—the beam search noising procedures we proposed alone are intended to yield convincing errors.

For each noising scheme, we took 100 $(X, Y)$ pairs from the development set (500 randomly chosen pairs combined), then generated $\tilde{Y}$ from $Y$. We then shuffled the examples and the order of $X$ and $\tilde{Y}$ such that the identity of $X$ and $\tilde{Y}$ *as well as the noising scheme* used to generate $\tilde{Y}$ were unknown[2]. Given $Y$, the task for human evaluators is to predict whether $X$ or $\tilde{Y}$ was the synthesized example. For every example, we had two separate evaluators label the sentence they thought was synthesized. We chose to do this labeling task ourselves (blind to system) since we were familiar with the noising schemes used to generate examples, which should reduce the number of misclassifications. Results are shown in Figure 4, and examples of the evaluation task are provided in Table 4.

### 5.2 Noise Frequency and Diversity

Comparing the performance using different noising methods on the CoNLL 2014 dataset to the human evaluation in the previous section, we see that generating errors which *match* the real distribution tends to result in higher performance, as seen by the poor performance of token noising relative

[2]Hence the human labelers cannot favor a particular scheme unless it can be distinguished from $\tilde{Y}$.

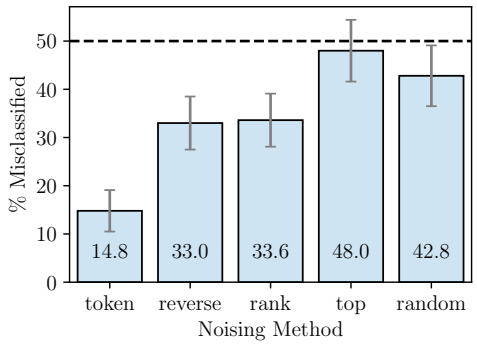


Figure 4: Percentage of time human evaluators misclassified synthesized noisy sentence $\tilde{Y}$ (vs. $X$) when using each noising scheme, along with 95% confidence intervals. The best we can expect any scheme to do is 50%.

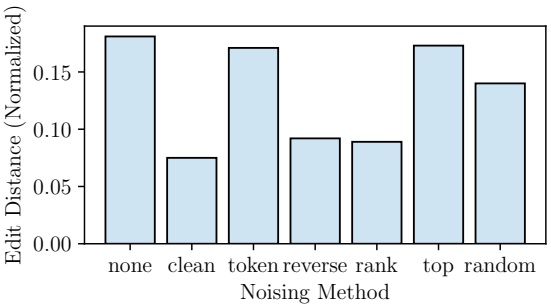


Figure 5: Mean edit distance between sentence pairs in $X$ and $Y$ *after* augmentation with noised sentences. *none* contains no synthesized examples while *clean* refers to the baseline of simply appending clean examples (source = target).

to the other methods. Injecting the appropriate *amount* of noise is important as well, as seen by improved performance when using beam search noising to increase diversity of outputs, and no performance gain when simply adding clean text.

We observe that token noising, despite matching the frequency of errors, fails to generate realistic errors (Figure 4). On the other hand, reverse noising yields significantly more convincing errors, but the edit distance between synthesized examples is significantly lower than in real data (Figure 5). A combination of sufficient amounts of noise and rich, diverse errors appears to lead to better model performance.

### 5.3 Error Type Distribution Mismatch

Mismatches in the distribution of error types can often severely impact the performance of data synthesis techniques for grammar correction (Felice, 2016). For example, only synthesizing noun number articles or preposition errors based on rules

| | Sentence | 1 or 2 |
|---|---|---|
| clean | Day after day , I get up at 8 o‘clock . | |
| 1 | I got up at 8 o‘clock day after day . | |
| 2 | Day after day , I get up 8 o‘clock in the week . | |
| clean | Thanks Giving Day in Korea is coming soon . | |
| 1 | In Korea , it ’s coming soon , thanks Giving day . | |
| 2 | Thanks Giving Day in korea is coming soon . | |
| clean | After I practiced , I could play the song perfectly . | |
| 1 | After the results , I could accomplish without a fault . | |
| 2 | When I tried that , I could play the song perfectly . | |
| clean | Currently , I ’m studying to take the TOEIC exam for my future career . | |
| 1 | I am studying to take TOEIC exam for career of my future . | |
| 2 | Currently , I will have take TOEIC exam for future career . | |
| clean | There is one child who is 15 years old and a mother who is around 50 . | |
| 1 | There are one child who is 15 years old and mother is around 50 . | |
| 2 | It has one child , 15 years old and the mother who is around 50 years old . | |
| clean | But at the beginning , I suffered from a horrible pain in my jaw . | |
| 1 | But at the first time , I suffer from a horrible pain on my jaw . | |
| 2 | But at the beginning , I suffered from a horrible pain in my jaw joint . | |

Table 4: Examples of nonsynthesized and synthesized sentences from validation set. Which example (1 or 2) was synthesized? Answers: 1, 1, 2, 1, 2, 1

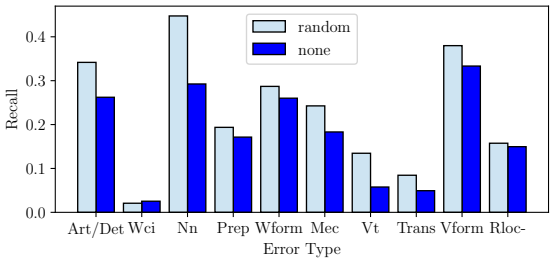


Figure 6: Recall vs. error type for the ten most frequent error types in our dev set. Noising improves recall uniformly across error types (See Ng et al. (2014) for a description of error types).

may improve the performance for those two error types, but may hurt overall performance. In contrast, the approaches we consider, with the exception of token noising, are fully data-driven, and hence we would expect gains across all different error types. We observe this is the case for random noising, as shown in Figure 6.

### 5.4 Data Sparsity and Domain Adaptation

Domain adaptation can yield significant differences in performance for dissimilar domains (such as those of the datasets used in our experiments) (Daumé III, 2009). The Lang-8, CoNLL, and JFLEG datasets contain online forum data and essay data from English learners. The $n$-gram language model is estimated using Common Crawl data from the web. The clean data which we noise is collected from a news corpus. Yet each dataset yields significant gains. This suggests that at current levels of system performance, data sparsity remains the key data issue, more so than domain adaptation.

It is also possible that LDC New York Times data is better matched to the CoNLL essay data than the Lang-8 forum data, and this in part accounts for the large gains we observe from training on synthesized data.

## 6 Conclusion

In this work, we address one of the key issues for developing translation-based grammar correction systems: the need for a large corpus of parallel data. We propose synthesizing parallel data by noising clean text, where instead of applying noise based on finite context windows, we instead train a reverse model and apply noise during the beam search procedure to synthesize noisy examples that human evaluators were nearly unable to distinguish from real examples. Our experiments suggest that the proposed data synthesis technique can yields almost as strong results as when training with additional nonsynthesized data. Hence, we hope that parallel data becomes less of a bottleneck, and more emphasis can be placed on developing better models that can capture the longer dependencies and structure in the text.

## Acknowledgments

We thank the anonymous reviewers for their helpful feedback, as well as Steven Tan for comments on an early draft.

## References

Mikel Artetxe, Gorka Labaka, Eneko Agirre, and Kyunghyun Cho. 2017. Unsupervised neural machine translation. *arXiv preprint arXiv:1710.11041* .

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

Samuel R Bowman, Luke Vilnis, Oriol Vinyals, Andrew M Dai, Rafal Jozefowicz, and Samy Bengio. 2015. Generating sentences from a continuous space. *arXiv preprint arXiv:1511.06349* .

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting esl errors using phrasal smt techniques. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, pages 249–256.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* .

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *arXiv preprint arXiv:1801.08831* .

Andrew M Dai and Quoc V Le. 2015. Semi-supervised sequence learning. In *Advances in Neural Information Processing Systems*. pages 3061–3069.

Hal Daumé III. 2009. Frustratingly easy domain adaptation. *arXiv preprint arXiv:0907.1815* .

Mariano Felice. 2016. Artificial error generation for translation-based grammatical error correction. Technical report, University of Cambridge, Computer Laboratory.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122* .

Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. 2014. Generative adversarial nets. In *Advances in neural information processing systems*. pages 2672–2680.

Kenneth Heafield. 2011. Kenlm: Faster and smaller language model queries. In *Proceedings of the Sixth Workshop on Statistical Machine Translation*. Association for Computational Linguistics, pages 187–197.

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

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *arXiv preprint arXiv:1605.06353* .

Nal Kalchbrenner, Lasse Espeholt, Karen Simonyan, Aaron van den Oord, Alex Graves, and Koray Kavukcuoglu. 2016. Neural machine translation in linear time. *arXiv preprint arXiv:1610.10099* .

Philipp Koehn and Rebecca Knowles. 2017. Six challenges for neural machine translation. *arXiv preprint arXiv:1706.03872* .

Ankit Kumar, Ozan Irsoy, Jonathan Su, James Bradbury, Robert English, Brian Pierce, Peter Ondruska, Ishaan Gulrajani, and Richard Socher. 2015. Ask me anything: Dynamic memory networks for natural language processing. *arXiv preprint arXiv:1506.07285* .

Guillaume Lample, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2017. Unsupervised machine translation using monolingual corpora only. *arXiv preprint arXiv:1711.00043* .

Jiwei Li, Michel Galley, Chris Brockett, Jianfeng Gao, and Bill Dolan. 2015. A diversity-promoting objective function for neural conversation models. *arXiv preprint arXiv:1510.03055* .

Jiwei Li, Will Monroe, and Dan Jurafsky. 2016. A simple, fast diverse decoding algorithm for neural generation. *arXiv preprint arXiv:1611.08562* .

Tal Linzen, Emmanuel Dupoux, and Yoav Goldberg. 2016. Assessing the ability of lstms to learn syntax-sensitive dependencies. *arXiv preprint arXiv:1611.01368* .

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 *International Joint Conference on Natural Language Processing (IJCNLP)*.

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*. Association for Computational Linguistics, Valencia, Spain,

pages 229–234. http://www.aclweb.org/anthology/E17-2037.

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.

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 (CoNLL-2013 Shared Task)*.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. *arXiv preprint arXiv:1707.05236* .

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Improving neural machine translation models with monolingual data. *arXiv preprint arXiv:1511.06709* .

Iulian V Serban, Alessandro Sordoni, Yoshua Bengio, Aaron Courville, and Joelle Pineau. 2015. Building end-to-end dialogue systems using generative hierarchical neural network models. *arXiv preprint arXiv:1507.04808* .

Alessandro Sordoni, Michel Galley, Michael Auli, Chris Brockett, Yangfeng Ji, Margaret Mitchell, Jian-Yun Nie, Jianfeng Gao, and Bill Dolan. 2015. A neural network approach to context-sensitive generation of conversational responses. *arXiv preprint arXiv:1506.06714* .

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

Pascal Vincent, Hugo Larochelle, Isabelle Lajoie, Yoshua Bengio, and Pierre-Antoine Manzagol. 2010. Stacked denoising autoencoders: Learning useful representations in a deep network with a local denoising criterion. *Journal of Machine Learning Research* 11(Dec):3371–3408.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, et al. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *arXiv preprint arXiv:1609.08144* .

Ziang Xie, Sida I Wang, Jiwei Li, Daniel Lévy, Aiming Nie, Dan Jurafsky, and Andrew Y Ng. 2017. Data noising as smoothing in neural network language models. *arXiv preprint arXiv:1703.02573* .

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