Empirical Error Modeling Improves Robustness of Noisy Neural Sequence Labeling

Marcin Namysl[1,2], **Sven Behnke**[1,2], and **Joachim Köhler**[1]

[1] Fraunhofer IAIS, Sankt Augustin, Germany
[2] Autonomous Intelligent Systems, University of Bonn, Germany
{Marcin.Namysl,Sven.Behnke,Joachim.Koehler}@iais.fraunhofer.de

Abstract

Despite recent advances, standard sequence labeling systems often fail when processing noisy user-generated text or consuming the output of an Optical Character Recognition (OCR) process. In this paper, we improve the noise-aware training method by proposing an empirical error generation approach that employs a sequence-to-sequence model trained to perform translation from error-free to erroneous text. Using an OCR engine, we generated a large parallel text corpus for training and produced several real-world noisy sequence labeling benchmarks for evaluation. Moreover, to overcome the data sparsity problem that exacerbates in the case of imperfect textual input, we learned noisy language model-based embeddings. Our approach outperformed the baseline noise generation and error correction techniques on the erroneous sequence labeling data sets. To facilitate future research on robustness, we make our code, embeddings, and data conversion scripts publicly available.

1 Introduction

Deep learning models have already surpassed human-level performance in many Natural Language Processing (NLP) tasks[1]. Sequence labeling systems have also reached extremely high accuracy (Akbik et al., 2019; Heinzerling and Strube, 2019). Still, NLP models often fail in scenarios, where non-standard text is given as input (Heigold et al., 2018; Belinkov and Bisk, 2018).

NLP algorithms are predominantly trained on error-free textual data but are also employed to process user-generated text (Baldwin et al., 2013; Derczynski et al., 2013) or consume the output of prior Optical Character Recognition (OCR) or Automatic Speech Recognition (ASR) processes (Miller et al., 2000). Errors that occur in any upstream component of an NLP system deteriorate the accuracy of the target downstream task (Alex and Burns, 2014).

[1]GLUE benchmark (Wang et al., 2018a): https://gluebenchmark.com/leaderboard

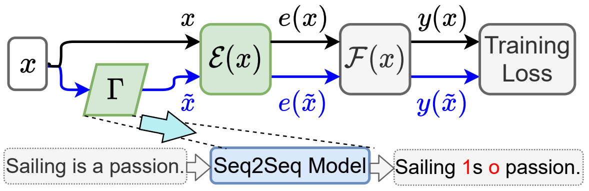


Figure 1: Our modification of the NAT approach (green boxes). We propose a learnable seq2seq-based error generator and re-train FLAIR embeddings using noisy text to improve the accuracy of noisy neural sequence labeling. Γ is a process that induces noise to the input x producing erroneous $\tilde{x}$. $\mathcal{E}(x)$ is an embedding matrix. $\mathcal{F}(x)$ is a sequence labeling model. $e(x)$ and $e(\tilde{x})$ are the embeddings of x and $\tilde{x}$, respectively. $y(x)$ and $y(\tilde{x})$ are the outputs of the model for x and $\tilde{x}$, respectively.

In this paper, we focus on the problem of performing sequence labeling on the text produced by an OCR engine. Moreover, we study the transferability of the methods learned to model OCR noise to the distribution of the human-generated errors. Both misrecognized and mistyped text pose a challenge for the standard models trained using error-free data (Namysl et al., 2020).

We make the following contributions (Figure 1):

- We propose a noise generation method for OCR that employs a sequence-to-sequence (seq2seq) model trained to translate from error-free to erroneous text (§4.1). Our approach improves the accuracy of noisy neural sequence labeling compared to prior work (§6.1).

- We present an unsupervised parallel training data generation method that utilizes an OCR engine (§4.2). Similarly, realistic noisy versions of popular sequence labeling data sets can be synthesized for evaluation (§5.5).

- We exploit erroneous text to perform Noisy Language Modeling (NLM; §4.5). Our NLM embeddings further improve the accuracy of noisy neural sequence labeling (§6.3), also in the case of the human-generated errors (§6.4).
- To facilitate future research on robustness, we integrate our methods into the Noise-Aware Training (NAT) framework (Namysl et al., 2020) and make our code, embeddings, and data conversion scripts publicly available.[2]

2 Related Work

Errors of OCR, ASR, and other text generators always pose a challenge to the downstream NLP systems (Lopresti, 2009; Packer et al., 2010; Ruiz et al., 2017). Hence, methods for improving robustness are becoming increasingly popular.

Data Augmentation A widely adopted method of providing robustness to non-standard input is to augment the training data with examples perturbed using a model that mimics the error distribution to be encountered at test time (Cubuk et al., 2019).

Apparently, the exact modeling of noise might be impractical or even impossible—thus, methods that employ randomized error patterns for training recently gained increasing popularity (Heigold et al., 2018; Lakshmi Narayan et al., 2019). Although trained using synthetic errors, these methods are often able to achieve moderate improvements on data from natural sources of noise (Belinkov and Bisk, 2018; Karpukhin et al., 2019).

Spelling- and OCR Post-correction The most widely used method of handling noisy text is to apply error correction on the input produced by human writers (*spelling correction*) or the output of an upstream OCR component (*OCR post-correction*).

A popular approach applies *monotone seq2seq* modeling for the correction task (Schnober et al., 2016). For instance, Hämäläinen and Hengchen (2019) proposed *Natas*—an OCR post-correction method that uses character-level Neural Machine Translation (NMT). They extracted parallel training data using embeddings learned from the erroneous text and used it as input to their translation model.

Grammatical Error Correction Grammatical Error Correction (GEC; Ng et al., 2013, 2014; Bryant et al., 2019) aims to automatically correct ungrammatical text. GEC can be approached as a translation from an ungrammatical to a grammatical language, which enabled NMT seq2seq models to be applied to this task (Yuan and Briscoe, 2016). Due to the limited size of human-annotated GEC corpora, NMT models could not be trained effectively (Lichtarge et al., 2019), though.

Several studies investigated generating realistic erroneous sentences from grammatically correct text to boost training data (Kasewa et al., 2018; Grundkiewicz et al., 2019; Choe et al., 2019; Qiu and Park, 2019). Inspired by *back-translation* (Sennrich et al., 2016; Edunov et al., 2018), Artificial Error Generation (AEG) approaches (Rei et al., 2017; Xie et al., 2018) train an intermediate model in reverse order—to translate correct sentences to erroneous ones. Following AEG, we generate a large corpus of clean and noisy sentences and train a seq2seq model to produce rich and diverse errors resembling the natural noise distribution (§3.3, 4.2).

Noise-Invariant Latent Representations Robustness can also be improved by encouraging the models to learn a similar latent representation for both the error-free and the erroneous input.

Zheng et al. (2016) introduced *stability training*—a general method used to stabilize predictions against small input perturbations. Piktus et al. (2019) proposed *Misspelling Oblivious Embeddings* that embed the misspelled words close to their error-free counterparts. Jones et al. (2020) developed *robust encodings* that balance stability (consistent predictions across various perturbations) and fidelity (accuracy on unperturbed input) by mapping sentences to a smaller discrete space of encodings. Although their model improved robustness against small perturbations, it decreased accuracy on the error-free input.

Recently, Namysl et al. (2020) proposed the Noise-Aware Training method that employs stability training and data augmentation objectives. They exploited both the error-free and the noisy samples for training and used a *confusion matrix-based* error model to imitate the errors. In contrast to their approach, we employ a more realistic empirical error distribution during training (§3.3) and observe improved accuracy at test time (§6.1).

3 Problem Definition

3.1 Noisy Neural Sequence Labeling

Namysl et al. (2020) pointed out that the standard NLP systems are generally trained using error-free

[2]https://github.com/mnamysl/nat-acl2021

textual input, which causes a discrepancy between the training and the test conditions. These systems are thus more susceptible to non-standard, corrupted, or adversarial input.

To model this phenomenon, they formulated the *noisy neural sequence labeling* problem, assuming that every input sentence might be subjected to some unknown token-level noising process $\Gamma = P(\tilde{x}_i|x_i)$, where x_i is the original i-th token, and $\tilde{x}_i$ is its distorted equivalent. As a solution, they proposed the NAT framework, which trains the sequence labeling model using auxiliary objectives that exploit both the original sentences and their copies corrupted using a noising process that imitates the naturally occurring errors (Figure 1).

3.2 Confusion Matrix-Based Error Model

Namysl et al. (2020) used a confusion matrix-based method to model insertions, deletions, and substitutions of characters. Given a corpus of paired noisy and manually corrected sentences $\mathcal{P}$, they estimated the natural error distribution by calculating the alignments between the pairs $(\tilde{x}, x) \in \mathcal{P}$ of noisy and clean sentences using the *Levenshtein distance* metric (Levenshtein, 1966).

Moreover, as $\mathcal{P}$ is usually laborious to obtain, they proposed a *vanilla error model*, which assumes that all types of edit operations are equally likely:

$$\sum_{\tilde{c} \in \Sigma \setminus \{\varepsilon\}} P_{ins}(\tilde{c}|\varepsilon) = P_{del}(\varepsilon|c) = \sum_{\tilde{c} \in \Sigma \setminus \{c, \varepsilon\}} P_{subst}(\tilde{c}|c),$$

where c and $\tilde{c}$ are the original and the perturbed characters, respectively, Σ is an alphabet, and ε is a symbol introduced to model insertion and deletions.

3.3 Realistic Empirical Error Modeling

Namysl et al. (2020) compared the NAT models that used the vanilla- and the empirically-estimated confusion matrix-based error model and observed no advantages of exploiting the test-time error distribution during training. *Would we make the same observation given a more realistic error model?*

Even though the methods that used randomized error patterns were often successful, we argue that leveraging the empirical noise distribution for training would be beneficial, providing additional accuracy improvements. The data produced by the naïve noise generation methods may not resemble naturally occurring errors, which could lead the downstream models to learn misleading patterns.

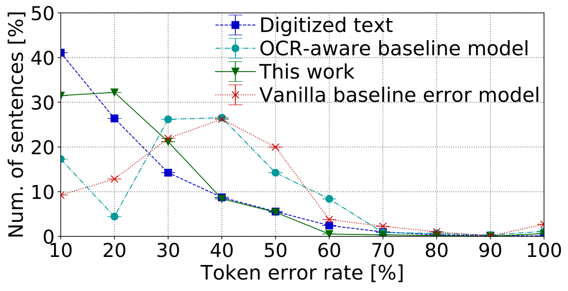


Figure 2: Distributions of the token error rates of sentences produced by the proposed and the baseline error models. For comparison, we plot the distribution of error rates in the text that contains naturally occurring errors. Each value n is the percentage of sentences with a token error rate in $[n-10, n)$.

In Figure 2, we compare the distributions of error rates of sentences produced by the proposed and the prior noise models with the distribution of errors in the digitized text. We can observe that the distribution of naturally occurring errors follows Zipf's law, while the baseline noise models produce Bell-shaped curves. Interestingly, both the vanilla and the empirical models exhibit similar characteristics, which could explain the observations from the prior work. In practice, the error rate is not uniform throughout the text. Some passages are recognized perfectly, while others can barely be deciphered. Our objective is thus to develop a noise model that produces a smoother distribution, imitating the errors encountered at test time more precisely (cf. *This work* in Figure 2).

Moreover, although the exact noise distribution in the test data cannot always be known beforehand, the noising process, e.g., an OCR engine, used to provide the input, can often be identified. We would thus take advantage of such prior knowledge to improve the efficiency of the downstream task.

3.4 Data Sparsity of Natural Language

Embeddings pre-trained on a large corpus of monolingual text are ubiquitous in NLP (Mikolov et al., 2013; Peters et al., 2018; Devlin et al., 2019). They capture syntactic and semantic textual features that can be exploited to solve higher-level NLP tasks.

Embeddings are generally trained using corpora that contain error-free text. Due to the data sparsity problem that arises from the large vocabulary sizes and the exponential number of feasible contexts, the majority of possible word sequences do not appear in the input data. Even though increasing the size of the training corpora was shown to improve the performance of language processing tasks (Brown et al., 2020), most of the misrecognized or mistyped tokens would still be unobserved and therefore poorly modeled when using the error-free text only. *Would it be beneficial to pre-train the embeddings on data that includes realistic erroneous sentences?*

3.5 The Flaws of Error Correction

Furthermore, we believe that the correction methods, although widely adopted, can only reliably manage moderately perturbed text (Flor et al., 2019). OCR post-correction has been reported to be challenging in the case of historical books that exhibit high OCR error rates (Rigaud et al., 2019).

We note that correction methods have no information about the downstream task to be performed. Moreover, in the automatic correction setting, they only provide the best guess for each token. Comparing their performance with the NAT approach in the context of sequence labeling would be informative.

4 Empirical Error Modeling

Figure 1 presents our modifications of the NAT framework. Firstly, we propose to replace the confusion matrix-based noising process (§3.2) with a noise induction method that generates a more realistic error distribution (§4.1-4.4). Secondly, to overcome the data sparsity problem (§3.4), we train language model-based embeddings using digitized text and use them as a substitution of the pre-trained model used in prior work (§4.5).

4.1 Sequence-to-Sequence Error Generator

Motivated by the AEG approaches (Rei et al., 2017; Xie et al., 2018), we propose a learnable error generation method that employs a character-level seq2seq model to perform monotone string translation (Schnober et al., 2016). It directly models the conditional probability $p(\tilde{x}|x)$ of mapping error-free text x into erroneous text $\tilde{x}$ using an attention-based encoder-decoder framework (Bahdanau et al., 2015). The encoder computes the representation $h{=}\{h_1, \ldots, h_n\}$ of x, where n is the length of x. The decoder generates $\tilde{x}$ one token at a time:

$$p(\tilde{x}|x) = \prod\nolimits_{i=1}^{n} p(\tilde{x}_i|\tilde{x}_{<i}, x, c),$$

where $c{=}f_{attn}(\{h_1, \ldots, h_n\})$ is a vector generated from h, and f_{attn} is an attention function.

Our models are trained to maximize the likelihood of the training data. At inference time, we randomly sample the subsequent tokens from the learned conditional language model.

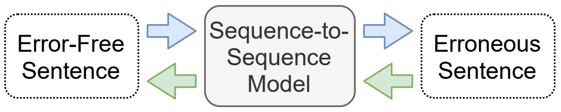


Figure 3: Schematic visualization of the error generation (blue arrows) and the error correction (green arrows) methods. The parallel data can be utilized to train seq2seq models for both tasks.

Note that our approach reverses the standard seq2seq error correction pipeline, which uses the erroneous text as input and trains the model to produce the corresponding error-free string (Figure 3). By interchanging the input and the output data, we can also readily train sentence correction models. One difference is that at inference time we would prefer to perform beam search and select the best decoding result rather than sampling subsequent characters from the learned distribution.

4.2 Unsupervised Parallel Data Generation

To train our error generation model (§4.1), we need a large parallel corpus $\mathcal{P}$ of error-free and erroneous sentences. AEG approaches use seed GEC corpora to learn the inverse models directly. Unfortunately, we are not aware of any comparably large resources for digitized text that could be used for this task.

To address this issue, we propose an unsupervised sentence-level parallel data generation approach for OCR (Figure 4). First, we collect a large seed corpus $\mathcal{T}$ that contains the error-free text. We then render each sentence and subsequently run text recognition on the rendered images using an OCR engine. Moreover, to increase the variation in training data, we sample different fonts for rendering. Furthermore, to simulate the distortions and degradation of the printed material, we induce pixel-level noise to the images before recognition.

Note that our approach is universal and could be used to generate parallel data sets for other tasks, e.g., an ASR system could be trained on samples from a Text-to-Speech engine (Wang et al., 2018b).

4.3 Sentence- and Word-Level Modeling

We note that the sequence labeling problem is formulated at the word-level, i.e., each word has a class label assigned to it. To employ our method in

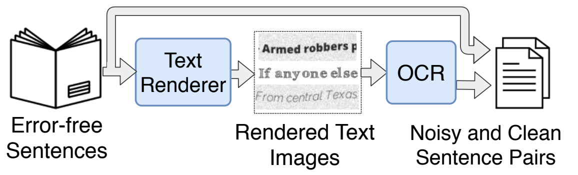


Figure 4: Our parallel data generation method for OCR. We render sentences extracted from a text corpus. Subsequently, an OCR engine recognizes the text depicted in the rendered images. Finally, the pairs of original and recognized sentences are gathered together to form a parallel corpus used to train translation models.

this scenario, we develop (i) a *sentence-level* and (ii) a *token-level* variant of our error generator.

Our sentence-level error generator uses a seq2seq model trained to translate from error-free to erroneous sentences. It can potentially utilize contextual information from surrounding words, which may improve the quality of the results. During the training of a NAT model, a learned seq2seq model translates the original input x to generate $\tilde{x}$. Subsequently, we use an alignment algorithm (§4.4) to transfer the word-level annotations from x to $\tilde{x}$.

Our token-level error generator uses a seq2seq model trained to translate from error-free to erroneous words. It relies exclusively on the input and the output words. We use the alignment algorithm to build a training set for this task, i.e., extract word-level parallel data from the corpus of parallel sentences (§4.2). During the training of a NAT model, a learned generator translates each word x_i from x to produce the erroneous sentence $\tilde{x}$.

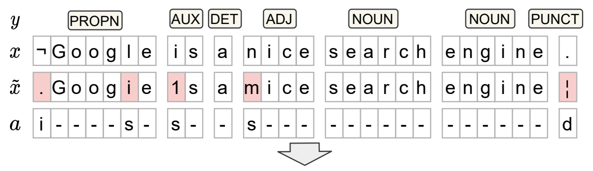
4.4 Word-Level Sentence Alignment

Figure 5 illustrates the alignment procedure, which we developed to extract word-level parallel training data for our token-level generator and to transfer the labels to the erroneous sentences for the sentence-level generator in the sequence labeling scenario.

To this end, we align each pair of error-free and noisy sentences at the word-level using the Levenshtein Distance algorithm. Our alignment procedure produces pairs of aligned words. The annotations for words are transferred accordingly.

4.5 Noisy Language Modeling

Recently, Xie et al. (2017) drew a connection between input noising in neural network language models and smoothing in n-gram models. We believe that data noising could be an effective technique for regularizing neural language models that could help to overcome the data sparsity problem of imperfect natural language text and enable learning meaningful representation of erroneous tokens.


y PROPN AUX DET ADJ NOUN NOUN PUNCT
x ¬ G o o g l e i s a n i c e s e a r c h e n g i n e .
x̃ . G o o g i e 1 s a m i c e s e a r c h e n g i n e ¦
a i - - - - s - s - - s - - - - - - - - - - - - - - - d

('Google', '.Googie', PROPN), ('is', '1s', AUX), ('a', 'a', DET), ('nice', 'mice', ADJ), ('search', 'search', NOUN), ('engine', 'engine', NOUN), ('.', '', PUNCT)

Figure 5: Our sentence alignment procedure. We align the original and the recognized sentences (x and $\tilde{x}$, respectively) using the sequence of edit operations a, which include insertions ”i”, deletions ”d”, and substitutions ”s” of characters. We use ”¬” and ”¦” as placeholders for the insertion and the deletion operation, respectively. Matched characters are marked with ”-”. The alignment procedure produces a list of paired error-free and possibly erroneous words with class labels (optional).

To this end, we propose to include the data from noisy sources in the corpora used to train LM-based embeddings. Specifically, in this work, we learn a noisy language model using the output of an OCR engine (§4.2) that captures the characteristics of OCR errors. Any other noisy source could be readily used to model related domains, e.g., ASR-transcripts or ungrammatical text.

5 Experimental Setup

5.1 Sequence-to-Sequence Error Generator

To learn our error generators (§4.1), we utilize the OpenNMT[3] toolkit (Klein et al., 2017).[4] We encode the input sentence at the character-level before feeding it to the seq2seq model. Subsequently, the output produced by the seq2seq model is decoded back to the original form (Figure 6).

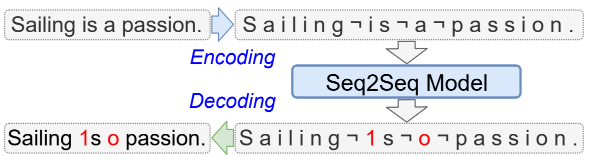

Sailing is a passion.
S a i l i n g ¬ i s ¬ a ¬ p a s s i o n .
Encoding
Seq2Seq Model
Decoding
Sailing 1s o passion.
S a i l i n g ¬ 1 s ¬ o ¬ p a s s i o n .

Figure 6: Sentence encoding-decoding schema. The whitespace characters are first replaced with a placeholder symbol ”¬”. The sentences are tokenized at the character-level by adding whitespace between every pair of characters. Decoding reverses this process.

[3]https://github.com/OpenNMT/OpenNMT-py

[4]We list all non-default hyper-parameters in Table 8.

5.2 Unsupervised Parallel Data Generation

Following the approach from §4.2, we generated a large parallel corpus $\mathcal{P}$ to train our error generation and correction models. We sampled 10 million sentences[5] from the English part of the *1 Billion Word Language Model Benchmark*[6] and used them as the source of error-free text, i.e., the seed corpus $\mathcal{T}$. We rendered each sentence as an image using the *Text Recognition Data Generator* package[7]. We used 90 different fonts for rendering and applied random distortions to the rendered images. Subsequently, we performed OCR on each image of text using a Python wrapper[8] for Tesseract-OCR[9] (Smith, 2007). We present the distribution of error rates in our noisy corpus in Figure 2 (cf. the *digitized text* plot).

5.3 Sequence Labeling

Training Setup We employed the NAT framework[10] (Figure 1) to study the robustness of sequence labeling systems. Following Akbik et al. (2018), we used a combination of *FLAIR* and *GloVe* embeddings in all experiments.[11] We employed the data augmentation ($\mathcal{L}_{\text{AUGM}}$) and the stability training ($\mathcal{L}_{\text{STAB}}$) objectives with default weights ($\alpha = 1.0$), as proposed by Namysl et al. (2020). Consistent with prior work, erroneous sentences $\tilde{x}$ were generated dynamically in every epoch.

Tasks We experimented with the Named Entity Recognition (NER) and Part-of-Speech Tagging (POST) tasks. NER aims to locate all named entity mentions in text and classify them into predefined classes, e.g., person names, locations, and organizations. POST is the process of tagging each word in the text with the corresponding part of speech.

Evaluation Setup The evaluation pipeline is shown in Figure 7. Following Akbik et al. (2018), we report the entity-level micro-average F1 score for NER and the accuracy for POST.

5.4 Baselines

Error Generation We compared our error generator with the OCR-aware noise model from Namysl et al. (2020). We used the noisy part of the parallel corpus $\mathcal{P}$ to estimate the confusion matrix employed by this baseline. Moreover, in the NLM experiment (§6.3), we also evaluated the vanilla error model proposed by Namysl et al. (2020).

Error Correction To evaluate error correction, we trained the sequence labeling models using the standard objective ($\mathcal{L}_0$) and employed the text correction method on the erroneous input before feeding it to the network (Figure 7).

We examined *Natas*[12], the seq2seq OCR post-correction method proposed by Hämäläinen and Hengchen (2019). We trained context-free error correction models compatible with Natas using our parallel corpus (§5.2). Moreover, we also employed the widely adopted spell checker *Hunspell*[13].

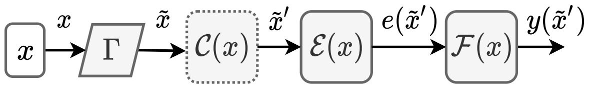


Figure 7: Evaluation pipeline. Γ is a noising process that transforms x into $\tilde{x}$. $\mathcal{C}(x)$ is an optional text correction module that returns $\tilde{x}'$ ($\tilde{x}'=\tilde{x}$, if $\mathcal{C}(x)$ is absent). $\mathcal{E}(x)$ is an embedding matrix. $\mathcal{F}(x)$ is a sequence labeling model. $e(\tilde{x}')$ and $y(\tilde{x}')$ are the embeddings and the output of the model for $\tilde{x}'$, respectively.

5.5 Data Sets

Original Benchmarks For NER, we employed the CoNLL 2003 data set (Tjong Kim Sang and De Meulder, 2003). To evaluate POST, we utilized the Universal Dependency Treebank (UD English EWT; Silveira et al., 2014). We present the detailed statistics of both data sets in Table 5.

Data Set	Geom. Distort.	Pixel-level Noise	CoNLL 2003	UD English EWT
Tesseract 3$^{\clubsuit}$	✗	✗	22.72%	23.31%
Tesseract 4$^{\diamondsuit}$	✓	✓	16.35%	22.12%
Tesseract 4$^{\heartsuit}$	✓	✗	14.89%	20.38%
Tesseract 4$^{\spadesuit}$	✗	✓	3.53%	5.83%
Typos	n/a	n/a	15.53%	15.22%

Table 1: The noisy sequence labeling data sets that we generated either by applying OCR on rendered sentences from an original benchmark (first four rows) or by inducing misspellings (last row). We generated multiple variants of the former data sets by combining geometrical distortions and pixel-level noise induction. The last two columns present the token error rates (the column headers indicate the names of the original benchmarks).

[5]Which accounts for about 253 million words.
[6]https://www.statmt.org/lm-benchmark
[7]https://pypi.org/project/trdg
[8]https://github.com/sirfz/tesserocr
[9]We used Tesseract v4.0 to generate the parallel data set.
[10]https://github.com/mnamysl/nat-acl2020
[11]Other hyper-parameters also follow Akbik et al. (2018).
[12]https://github.com/mikahama/natas
[13]https://hunspell.github.io

Training Loss	Noise Model	Correction Method	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
	n/a	—	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
$\mathcal{L}_0$	n/a	Hunspell	**92.54±0.08**	**82.17±0.11**	**85.80±0.11**	**86.70±0.07**	**91.73±0.11**
	n/a	Natas	**92.54±0.08**	77.80±0.19	84.50±0.11	85.24±0.10	91.33±0.13
	confusion matrix (§3.2)	—	92.56±0.06	**85.29±0.16**	88.62±0.08	89.19±0.12	92.04±0.07
$\mathcal{L}_{\text{AUGM}}$	seq2seq (token-level, §4.3)	—	**92.76±0.07**	**85.38±0.16**	**89.39±0.17**	**89.99±0.22**	**92.37±0.10**
	seq2seq (sentence-level, §4.3)	—	**92.81±0.11**	84.38±0.15	88.96±0.18	**89.67±0.26**	**92.44±0.17**
	confusion matrix (§3.2)	—	92.23±0.12	**84.49±0.10**	87.58±0.13	88.40±0.20	91.65±0.14
$\mathcal{L}_{\text{STAB}}$	seq2seq (token-level, §4.3)	—	**92.24±0.18**	**84.25±0.23**	**88.24±0.25**	**88.91±0.21**	**91.86±0.16**
	seq2seq (sentence-level, §4.3)	—	**92.45±0.12**	83.89±0.30	**88.14±0.23**	**88.88±0.11**	**91.99±0.11**

(a) English CoNLL 2003

Training Loss	Noise Model	Correction Method	Original Data	Tesseract 3$^{\clubsuit}$	Tesseract 4$^{\diamondsuit}$	Tesseract 4$^{\heartsuit}$	Tesseract 4$^{\spadesuit}$
	n/a	—	**96.96±0.04**	86.75±0.16	86.97±0.14	88.30±0.16	94.34±0.07
$\mathcal{L}_0$	n/a	Hunspell	**96.96±0.04**	87.53±0.14	86.74±0.14	88.12±0.16	94.49±0.08
	n/a	Natas	**96.96±0.04**	**88.98±0.10**	**88.94±0.14**	**89.68±0.16**	**95.11±0.08**
	confusion matrix (§3.2)	—	**96.90±0.06**	**91.35±0.13**	92.12±0.14	92.99±0.21	96.17±0.07
$\mathcal{L}_{\text{AUGM}}$	seq2seq (token-level, §4.3)	—	96.76±0.04	**91.44±0.11**	**93.65±0.13**	**94.19±0.10**	**96.26±0.07**
	seq2seq (sentence-level, §4.3)	—	96.78±0.06	90.92±0.08	93.37±0.08	**94.10±0.03**	**96.27±0.03**
	confusion matrix (§3.2)	—	**96.80±0.04**	91.16±0.07	91.93±0.11	92.77±0.10	96.06±0.02
$\mathcal{L}_{\text{STAB}}$	seq2seq (token-level, §4.3)	—	96.65±0.07	**91.36±0.12**	**93.34±0.09**	**93.97±0.05**	**96.14±0.07**
	seq2seq (sentence-level, §4.3)	—	96.67±0.05	90.70±0.14	93.05±0.17	93.71±0.13	**96.15±0.05**

(b) UD English EWT

Table 2: Comparison of error generation (§6.1) and error correction (§6.2) approaches on the original and noisy English CoNLL 2003 and the UD English EWT test sets (§5.5). We report mean and standard deviation F1 scores (CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

Noisy Benchmarks Unfortunately, we did not find any publicly available noisy sequence labeling data set that could be used to benchmark different methods for improving robustness. To this end, we generated several noisy versions of the original sequence labeling data sets (Table 1). We extracted the sentences from each original benchmark and applied the procedure described in §4.2.[14] We transferred the word-level annotations as described in §4.4. Finally, we produced the data in the CoNLL format (Table 7).

Moreover, to evaluate the transferability of error generators, we followed Namysl et al. (2020) and synthetically induced misspellings to the error-free data sets. To this end, we used the lookup tables of possible lexical replacements released by Belinkov and Bisk (2018) and Piktus et al. (2019).[15]

6 Experimental Results

6.1 Empirical Noise Generation Approaches

In this experiment, we compared the NAT models that employed either our seq2seq noise generators or the baseline error models (Table 2). In this evaluation scenario, we do not employ $\mathcal{C}(x)$ (Figure 7).

Our error generators outperformed the OCR-aware confusion matrix-based model on the noisy benchmarks generated using the Tesseract 4 engine. The advantage of our method was less emphasized in the case of the *Tesseract 3*$^{\clubsuit}$ data sets. The token-level translation method performed better than the sentence-level variant, while the latter was more efficient when the error rate of the input was lower (cf. the *original data* and the *Tesseract 4*$^{\spadesuit}$ columns), although it often struggled with translating long sentences. Moreover, data augmentation generally outperformed stability training, which is consistent with the observation from Namysl et al. (2020).

Furthermore, we observe a slight decrease in accuracy on the original UD English EWT with both auxiliary objectives. We believe that this was caused by the different proportions of the tokens that were perturbed during training by our seq2seq error generators (e.g., 18% and 19.5% in the case of our token-level model for CoNLL2003 and UD English EWT, respectively). The trade-off between accuracy for clean and noisy data has thus been shifted towards the latter. We also notice a greater advantage of the seq2seq method over the baseline

[14] We directly applied both Tesseract v3.04 and v4.0. We used different sets of distortions and image backgrounds than those employed to generate parallel training data.

[15] We merged both sets of misspellings for evaluation.

Training Loss	Noise Model	NLM	Original Data	Tesseract 3$^\clubsuit$	Tesseract 4$^\diamondsuit$	Tesseract 4$^\heartsuit$	Tesseract 4$^\spadesuit$
$\mathcal{L}_0$	n/a	✗	**92.54±0.08**	80.48±0.09	84.71±0.19	85.62±0.08	91.50±0.08
	n/a	✓	92.09±0.07	**83.83±0.21**	**88.17±0.12**	**88.71±0.17**	**91.68±0.09**
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	✗	**92.56±0.06**	85.29±0.16	88.62±0.08	89.19±0.12	92.04±0.07
	vanilla (§3.2)	✗	92.39±0.11	85.59±0.23	88.01±0.17	88.65±0.20	91.93±0.13
	vanilla (§3.2)	✓	92.45±0.05	**87.28±0.19**	**90.12±0.19**	**90.43±0.19**	**92.17±0.05**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	✗	**92.23±0.12**	84.49±0.10	87.58±0.13	88.40±0.20	**91.65±0.14**
	vanilla (§3.2)	✗	92.04±0.06	84.63±0.17	87.24±0.24	88.02±0.10	**91.52±0.12**
	vanilla (§3.2)	✓	91.85±0.07	**86.79±0.11**	**89.32±0.12**	**89.77±0.05**	**91.51±0.07**

Table 3: Comparison of the NAT approach with and without our NLM embeddings (§6.3) on the English CoNLL 2003 test set (§5.5). We report mean and standard deviation F1 scores over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). The *NLM* column indicates whether the model employed our NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

on the noisy UD English EWT data sets.

Additionally, in §A, we analyze the relationship between the size of the parallel corpus used for training and the F1 score of the NER task.

6.2 Error Generation vs. Error Correction

We compared the NAT approach with the baseline correction methods (§5.4). Preliminary experiments revealed that these baselines underperformed due to the *overcorrection* problem. To make them more competitive, we extended their default dictionaries by adding all tokens from the corresponding test sets for evaluation. Although the vocabulary of a test set could rarely be entirely determined, this setting would simulate a scenario where accurate in-domain vocabularies could be exploited.

Table 2 includes the results of this experiment. As expected, although more general, error correction techniques were outperformed by the NAT approach regardless of the noising method used. Surprisingly, Hunspell performed better than Natas on CoNLL 2003. We carried out a thorough inspection of the results of both methods and found out that Natas, although generally more accurate, had problems with recognizing tokens that were a part of entities. This behavior could be a flaw of data-driven error correction methods, as the entities are relatively rare in written text and are often out-of-vocabulary tokens (Alex and Burns, 2014).

6.3 Noisy Language Modeling

FLAIR (Akbik et al., 2018) learns a bidirectional LM to represent sequences of characters. We used the target side of our parallel data corpus (§5.2) to re-train FLAIR embeddings on the noisy digitized text.[16] Subsequently, we compared the accuracy of the vanilla NAT models (§3.2) that employed either the pre-trained or our NLM embeddings. Moreover, we do not use $\mathcal{C}(x)$ in this scenario (Figure 7).

Note that the noise model and the embeddings are two distinct components of the NAT architecture (Γ and $\mathcal{E}(x)$ in Figure 1, respectively) and therefore they could be easily combined. However, in this work, we do not mix our NLM with empirically estimated error models to avoid the twofold empirical error modeling effect. We leave the evaluation of this combination to future work.

Table 3 summarizes the results of this experiment. Our method significantly improved the accuracy across all training objectives, even when we employed exclusively the standard training objective for the sequence labeling task ($\mathcal{L}_0$). Surprisingly, we also achieved evident improvements for the noisy data set generated using the Tesseract 3 engine, which confirms that NLM embeddings can model the features of erroneous tokens even in the out-of-domain scenarios. On the other hand, the NLM slightly decreased the accuracy on the original data for the standard training objective. We plan to investigate this effect in future work by eliminating possible differences in the pre-training procedure and comparing our NLM against a model trained on the original error-free text corpus instead of using the embeddings from Akbik et al. (2018).

6.4 Human-Generated Errors

In this experiment, we evaluated the utility of our seq2seq error generators learned to model OCR noise (§6.1) and our NLM embeddings (§6.3) in a scenario where the input contains human-generated

[16]The hyper-parameters were consistent with prior work.

Training Loss	Noise Model	NLM	English CoNLL 2003
$\mathcal{L}_0$	n/a	✗	88.79±0.07
	n/a	✓	**89.60±0.24**
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	✗	90.82±0.12
	vanilla (§3.2)	✗	90.77±0.14
	vanilla (§3.2)	✓	**91.10±0.05**
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	✗	90.30±0.13
	vanilla (§3.2)	✗	90.34±0.06
	vanilla (§3.2)	✓	**90.53±0.07**

(a) NLM Embeddings

Training Loss	Noise Model	English CoNLL 2003	UD English EWT
$\mathcal{L}_0$	n/a	88.79±0.07	90.54±0.11
$\mathcal{L}_{\text{AUGM}}$	confusion matrix (§3.2)	**90.82±0.12**	**93.63±0.11**
	seq2seq (token-level, §4.3)	**90.92±0.13**	92.87±0.08
	seq2seq (sentence-level, §4.3)	**90.77±0.19**	92.68±0.09
$\mathcal{L}_{\text{STAB}}$	confusion matrix (§3.2)	**90.30±0.13**	**93.37±0.05**
	seq2seq (token-level, §4.3)	**90.19±0.12**	92.79±0.08
	seq2seq (sentence-level, §4.3)	**90.15±0.16**	92.42±0.11

(b) Empirical Error Generation Methods

Table 4: Transferability of the methods learned to model OCR noise to the distribution of the human-generated errors (§6.4): **(a)** Comparison of the NAT approach with and without our NLM embeddings on the English CoNLL 2003 test set with human-generated errors. **(b)** Comparison of empirical error generation approaches on the English CoNLL 2003 and the UD English EWT test sets with human-generated errors. We report mean and standard deviation F1 scores (CoNLL 2003) and accuracies (UD English EWT) over five runs with different random initialization. $\mathcal{L}_0$, $\mathcal{L}_{\text{AUGM}}$, $\mathcal{L}_{\text{STAB}}$ is the standard, the data augmentation, and the stability objective, respectively (Namysl et al., 2020). The *NLM* column indicates whether the model employed our NLM embeddings. **Bold** values indicate top results (within the models trained using the same objective) that are statistically inseparable (Welch's t-test; $p < 0.05$).

errors. For evaluation, we used the noisy data sets with synthetically induced misspellings (§5.5). We do not employ $\mathcal{C}(x)$ in this scenario (Figure 7).

Table 4 summarizes the results of this experiment. The models with our NLM embeddings outperformed the baselines for all training objectives (Table 4a). The seq2seq error generation approach performed on par with the confusion matrix-based models on the CoNLL 2003 data set, while the latter achieved better accuracy on the UD English EWT data set (Table 4b).

We believe that this difference was caused by the discrepancy between the data distributions. Note that although the data used in this experiment reflects the patterns of human-generated errors, the distribution of these errors does not necessarily follow the natural distribution of human-generated errors, as it was synthetically generated using a fixed replacement probability that was uniform across all candidates.[17] Nevertheless, our methods proved to be beneficial in this scenario, which would suggest that the errors made by human writers and by the text recognition engines have common characteristics that were exploited by our method.

7 Conclusions

In this work, we studied the task of performing sequence labeling on noisy digitized and human-generated text. We extended the NAT approach and proposed the empirical error generator that performs the translation from error-free to erroneous text (§4.1). To train our generator, we developed an unsupervised parallel data synthesis method (§4.2). Analogously, we produced several realistic noisy evaluation benchmarks (§5.5). Moreover, we introduced the NLM embeddings (§4.5) that overcome the data sparsity problem of natural language.

Our approach outperformed the baseline noise induction and error correction methods, improving the accuracy of the noisy neural sequence labeling task (§6.1-6.3). Furthermore, we demonstrated that our methods are transferable to the out-of-domain scenarios - human-generated errors (§6.4) and the noise induced by a different OCR engine (§6.1, 6.3). We incorporated our approach into the NAT framework and make the code, embeddings, and scripts from our experiments publicly available.

Grundkiewicz and Junczys-Dowmunt (2019) showed that that unsupervised systems benefit from domain adaptation on authentic labeled data. For future work, we plan to fine-tune NAT models pretrained on synthetic samples using the labeled data generated directly by the noising process.

[17]For comparison, we visualized the error distributions of our noisy benchmarks in Figure 9.

References

Alan Akbik, Tanja Bergmann, and Roland Vollgraf. 2019. Pooled contextualized embeddings for named entity recognition. 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 724–728, Minneapolis, Minnesota. Association for Computational Linguistics.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649, Santa Fe, New Mexico, USA. Association for Computational Linguistics.

Beatrice Alex and John Burns. 2014. Estimating and rating the quality of optically character recognised text. In *Proceedings of the First International Conference on Digital Access to Textual Cultural Heritage*, DATeCH '14, pages 97–102, New York, NY, USA. ACM.

Dzmitry Bahdanau, Kyung Hyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. 3rd International Conference on Learning Representations, ICLR 2015 ; Conference date: 07-05-2015 Through 09-05-2015.

Timothy Baldwin, Paul Cook, Marco Lui, Andrew MacKinlay, and Li Wang. 2013. How noisy social media text, how diffrnt social media sources? In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 356–364, Nagoya, Japan. Asian Federation of Natural Language Processing.

Yonatan Belinkov and Yonatan Bisk. 2018. Synthetic and natural noise both break neural machine translation. In *6th International Conference on Learning Representations, ICLR 2018, Vancouver, BC, Canada, April 30 - May 3, 2018, Conference Track Proceedings*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

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.

Yo Joong Choe, Jiyeon Ham, Kyubyong Park, and Yeoil Yoon. 2019. A neural grammatical error correction system built on better pre-training and sequential transfer learning. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 213–227, Florence, Italy. Association for Computational Linguistics.

Ekin D. Cubuk, Barret Zoph, Dandelion Mane, Vijay Vasudevan, and Quoc V. Le. 2019. AutoAugment: Learning augmentation strategies from data. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proceedings of the International Conference Recent Advances in Natural Language Processing RANLP 2013*, pages 198–206, Hissar, Bulgaria. INCOMA Ltd. Shoumen, BULGARIA.

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.

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*, pages 489–500, Brussels, Belgium. Association for Computational Linguistics.

Michael Flor, Michael Fried, and Alla Rozovskaya. 2019. A benchmark corpus of English misspellings and a minimally-supervised model for spelling correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 76–86, Florence, Italy. Association for Computational Linguistics.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2019. Minimally-augmented grammatical error correction. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 357–363, Hong Kong, China. Association for Computational Linguistics.

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*, pages 252–263, Florence, Italy. Association for Computational Linguistics.

Mika Hämäläinen and Simon Hengchen. 2019. From the paft to the fiiture: A fully automatic NMT and word embeddings method for OCR post-correction. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 431–436, Varna, Bulgaria. INCOMA Ltd.

Georg Heigold, Stalin Varanasi, Günter Neumann, and Josef van Genabith. 2018. How robust are character-based word embeddings in tagging and MT against wrod scramlbing or randdm nouse? In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Track)*, pages 68–80, Boston, MA. Association for Machine Translation in the Americas.

Benjamin Heinzerling and Michael Strube. 2019. Sequence tagging with contextual and non-contextual subword representations: A multilingual evaluation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 273–291, Florence, Italy. Association for Computational Linguistics.

Erik Jones, Robin Jia, Aditi Raghunathan, and Percy Liang. 2020. Robust encodings: A framework for combating adversarial typos. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2752–2765, Online. Association for Computational Linguistics.

Vladimir Karpukhin, Omer Levy, Jacob Eisenstein, and Marjan Ghazvininejad. 2019. Training on synthetic noise improves robustness to natural noise in machine translation. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 42–47, Hong Kong, China. Association for Computational Linguistics.

Sudhanshu Kasewa, Pontus Stenetorp, and Sebastian Riedel. 2018. Wronging a right: Generating better errors to improve grammatical error detection. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 4977–4983, Brussels, Belgium. Association for Computational Linguistics.

Guillaume Klein, Yoon Kim, Yuntian Deng, Jean Senellart, and Alexander Rush. 2017. OpenNMT: Open-source toolkit for neural machine translation. In *Proceedings of ACL 2017, System Demonstrations*, pages 67–72, Vancouver, Canada. Association for Computational Linguistics.

Pooja Lakshmi Narayan, Ajay Nagesh, and Mihai Surdeanu. 2019. Exploration of noise strategies in semi-supervised named entity classification. In *Proceedings of the Eighth Joint Conference on Lexical and Computational Semantics (*SEM 2019)*, pages 186–191, Minneapolis, Minnesota. Association for Computational Linguistics.

Vladimir Iosifovich Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady*, 10(8).

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: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Daniel Lopresti. 2009. Optical character recognition errors and their effects on natural language processing. *International Journal on Document Analysis and Recognition (IJDAR)*, 12(3):141–151.

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.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, page 3111–3119, Red Hook, NY, USA. Curran Associates Inc.

David Miller, Sean Boisen, Richard Schwartz, Rebecca Stone, and Ralph Weischedel. 2000. Named entity extraction from noisy input: Speech and OCR. In *Sixth Applied Natural Language Processing Conference*, pages 316–324, Seattle, Washington, USA. Association for Computational Linguistics.

Marcin Namysl, Sven Behnke, and Joachim Köhler. 2020. NAT: Noise-aware training for robust neural sequence labeling. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 1501–1517, Online. 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.

Thomas L. Packer, Joshua F. Lutes, Aaron P. Stewart, David W. Embley, Eric K. Ringger, Kevin D. Seppi, and Lee S. Jensen. 2010. Extracting person names from diverse and noisy ocr text. In *Proceedings of the Fourth Workshop on Analytics for Noisy Unstructured Text Data*, AND '10, page 19–26, New York, NY, USA. Association for Computing Machinery.

Matthew Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. 2018. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Associ-*

ation for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 2227–2237, New Orleans, Louisiana. Association for Computational Linguistics.

Aleksandra Piktus, Necati Bora Edizel, Piotr Bojanowski, Edouard Grave, Rui Ferreira, and Fabrizio Silvestri. 2019. Misspelling oblivious word embeddings. 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 3226–3234, Minneapolis, Minnesota. Association for Computational Linguistics.

Mengyang Qiu and Jungyeul Park. 2019. Artificial error generation with fluency filtering. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 87–91, Florence, Italy. Association for Computational Linguistics.

Marek Rei, Mariano Felice, Zheng Yuan, and Ted Briscoe. 2017. Artificial error generation with machine translation and syntactic patterns. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 287–292, Copenhagen, Denmark. Association for Computational Linguistics.

C. Rigaud, A. Doucet, M. Coustaty, and J. Moreux. 2019. ICDAR 2019 competition on post-OCR text correction. In *2019 International Conference on Document Analysis and Recognition (ICDAR)*, pages 1588–1593.

Nicholas Ruiz, Mattia Antonino Di Gangi, Nicola Bertoldi, and Marcello Federico. 2017. Assessing the tolerance of neural machine translation systems against speech recognition errors. In *Interspeech 2017, 18th Annual Conference of the International Speech Communication Association, Stockholm, Sweden, August 20-24, 2017*, pages 2635–2639.

Carsten Schnober, Steffen Eger, Erik-Lân Do Dinh, and Iryna Gurevych. 2016. Still not there? Comparing traditional sequence-to-sequence models to encoder-decoder neural networks on monotone string translation tasks. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1703–1714, Osaka, Japan. The COLING 2016 Organizing Committee.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Improving neural machine translation models with monolingual data. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 86–96, Berlin, Germany. Association for Computational Linguistics.

Natalia Silveira, Timothy Dozat, Marie-Catherine de Marneffe, Samuel Bowman, Miriam Connor, John Bauer, and Chris Manning. 2014. A gold standard dependency corpus for English. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC'14)*, pages 2897–2904, Reykjavik, Iceland. European Language Resources Association (ELRA).

Ray Smith. 2007. An overview of the Tesseract OCR engine. In *Proceedings of the Ninth International Conference on Document Analysis and Recognition (ICDAR 2007)*, volume 2, pages 629–633.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147.

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

Dingmin Wang, Yan Song, Jing Li, Jialong Han, and Haisong Zhang. 2018b. A hybrid approach to automatic corpus generation for Chinese spelling check. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2517–2527, Brussels, Belgium. Association for Computational Linguistics.

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: Human Language Technologies, Volume 1 (Long Papers)*, pages 619–628, New Orleans, Louisiana. Association for Computational Linguistics.

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. In *International Conference on Learning Representations (ICLR)*.

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

Stephan Zheng, Yang Song, Thomas Leung, and Ian J. Goodfellow. 2016. Improving the robustness of deep neural networks via stability training. In *2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2016, Las Vegas, NV, USA, June 27-30, 2016*, pages 4480–4488.

A Relationship with the Corpus Size

Empirical error generators are especially beneficial when we can approximate the noise distribution to be encountered at test time. In this experiment, we aimed to answer the question, how much parallel training data is required to train a solid seq2seq error generation model.

Figure 8 shows that the NAT models that used our seq2seq error generator performed better than those employing the baseline vanilla error model proposed by Namysl et al. (2020) for all noisy benchmarks that were generated using the *Tesseract 4* OCR engine. The improvements were observed even when we used as few as 1000 parallel training sentences. Our method also outperformed the baseline on the original CoNLL 2003 benchmark. On the contrary, the accuracy of models trained using our generator fell slightly behind the baseline on the *Tesseract 3*♣ and *Typos* data sets.

B Sequence Labeling Data Sets

Original Benchmarks Table 5 presents the detailed statistics of the original sequence labeling benchmarks used in our experiments. For NER, we employed CoNLL 2003[18] (Tjong Kim Sang and De Meulder, 2003). To evaluate POST, we utilized Universal Dependency Treebank (UD English EWT[19]; Silveira et al., 2014).

Noisy Benchmarks Table 6 presents the error rates and the correction accuracies of the Natas and Hunspell methods calculated on the test sets of the noisy sequence labeling benchmarks. Moreover, Table 7 shows an excerpt from a noisy sequence labeling data set generated for evaluation.

Furthermore, Figure 9 presents the distribution of token error rates in relation to the percentage number of sentences in our noisy data sets. For comparison, we also included the distributions obtained by applying different noise generation methods - the vanilla- and the OCR-aware confusion matrix-based models by Namysl et al. (2020), and our token-level seq2seq error generator.

We note that the error distribution of our noisy data sets is closer to the Zipf distribution in contrast to the results of prior methods that exhibit a Bell-Curve pattern. Note that the *Typos* data set was generated by randomly sampling possible lexical replacement candidates from the lookup tables, hence its distribution exhibits slightly different characteristics than the noisy data sets generated by directly applying the OCR engine to the rendered text images. Based on the above results,

[18] https://www.clips.uantwerpen.be/conll2003/ner

[19] https://universaldependencies.org/treebanks/en_ewt (version 2.6)

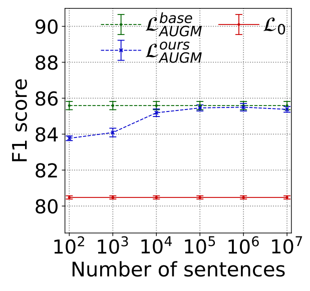
	Train	Dev	Test	Total
Sentences	14,041	3,250	3,453	20744
Tokens	203,621	51,362	46,435	301418
PER	6,600	1,842	1,617	10059
LOC	7,140	1,837	1,668	10645
ORG	6,321	1,341	1,661	9323
MISC	3,438	922	702	5062

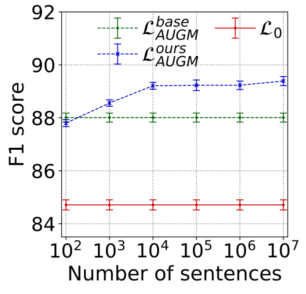
(a) English CoNLL 2003.

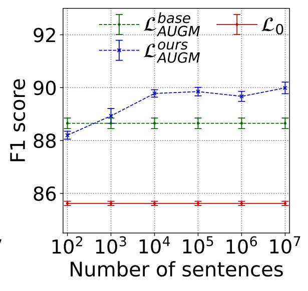
	Train	Dev	Test	Total
Sentences	12543	2002	2077	16622
Tokens	204585	25148	25096	254829
ADJ	12458	1784	1689	15931
ADP	17625	2021	2020	21666
ADV	10553	1264	1226	13043
AUX	12396	1512	1504	15412
CCONJ	6703	780	738	8221
DET	16284	1895	1896	20075
INTJ	688	115	120	923
NOUN	34765	4196	4129	43090
NUM	3996	378	536	4910
PART	5567	630	630	6827
PRON	18584	2219	2158	22961
PROPN	12945	1879	2075	16899
PUNCT	23676	3083	3106	29865
SCONJ	3850	403	386	4639
SYM	643	75	100	818
VERB	23005	2759	2644	28408
X	847	155	139	1141

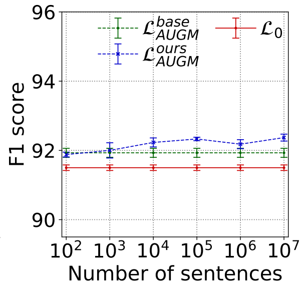
(b) UD English EWT.

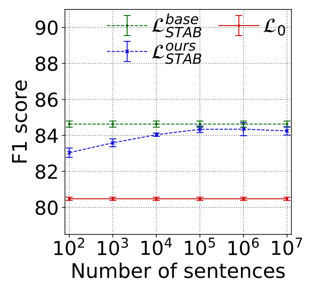
Table 5: Statistics of the English CoNLL 2003 and the UD English EWT data sets. We present statistics of the training (Train) development (Dev) and test (Test) sets, including the number of sentences and tokens. CoNLL 2003 contains the annotations for the following entity types: person names (PER), locations (LOC), organizations (ORG), and miscellaneous (MISC). For UD English EWT, the following universal POS tags were included: ADJ (adjective), ADP (adposition), ADV (adverb), AUX (auxiliary), CCONJ (coordinating conjunction), DET (determiner), INTJ (interjection), NOUN (noun), NUM (numeral), PART (particle), PRON (pronoun), PROPN (proper noun), PUNCT (punctuation), SCONJ (subordinating conjunction), SYM (symbol), VERB (verb), X (other).

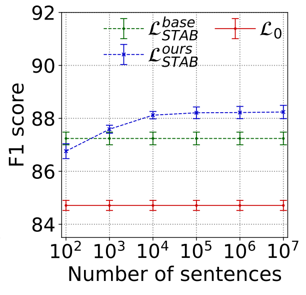

(a) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $3^{\clubsuit}$).

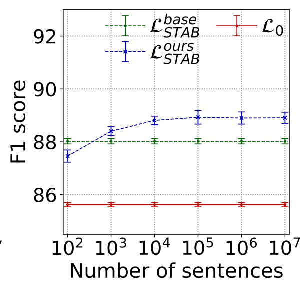

(b) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\diamondsuit}$).

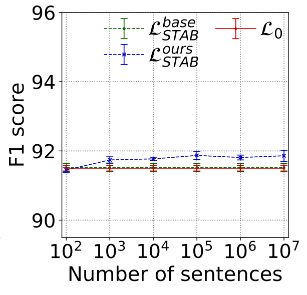

(c) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\heartsuit}$).

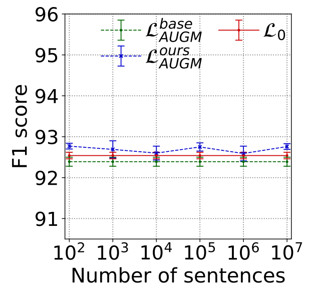

(d) $\mathcal{L}_{\text{AUGM}}$ (Tesseract $4^{\spadesuit}$).

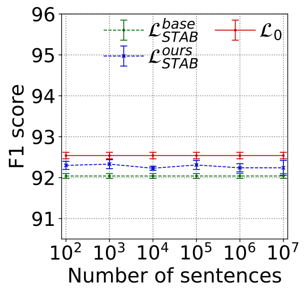

(e) $\mathcal{L}_{\text{STAB}}$ (Tesseract $3^{\clubsuit}$).

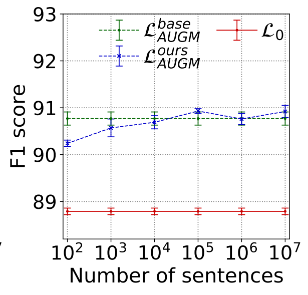

(f) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\diamondsuit}$).


(g) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\heartsuit}$).


(h) $\mathcal{L}_{\text{STAB}}$ (Tesseract $4^{\spadesuit}$).


(i) $\mathcal{L}_{\text{AUGM}}$ (original data set).


(j) $\mathcal{L}_{\text{STAB}}$ (original data set).


(k) $\mathcal{L}_{\text{AUGM}}$ (Typos).

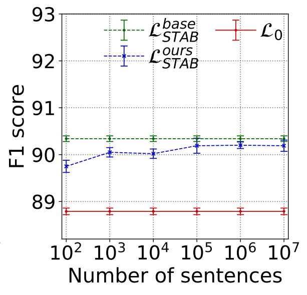

(l) $\mathcal{L}_{\text{STAB}}$ (Typos).

Figure 8: F1 score in relation to the number of parallel sentences. The experiments were conducted on the original English CoNLL 2003 benchmark and its noisy variants: Tesseract $3^{\clubsuit}$, Tesseract $4^{\diamondsuit}$, Tesseract $4^{\heartsuit}$, Tesseract $4^{\spadesuit}$, and Typos. We compare the accuracy of our token-level seq2seq approach with the vanilla error model (Namysl et al., 2020), and the standard objective ($\mathcal{L}_0$). We present the results for both auxiliary objectives: the data augmentation ($\mathcal{L}^{\text{ours}}_{\text{AUGM}}$, $\mathcal{L}^{\text{base}}_{\text{AUGM}}$) and the stability training ($\mathcal{L}^{\text{ours}}_{\text{STAB}}$, $\mathcal{L}^{\text{base}}_{\text{STAB}}$).

we believe that our noisy data sets are better suited for the evaluation of the robustness of sequence labeling models than the data generated by the prior approaches.

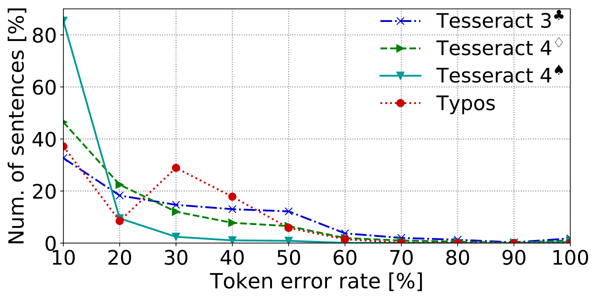
Data Conversion Scripts Because of licensing and copyright reasons, we did not submit the noisy data sets directly. Our code includes the scripts for the conversion of the original benchmarks into their noisy variants. For reference, we added excerpts of the noisy UD English EWT data set in the supplementary materials.

C Reproducibility

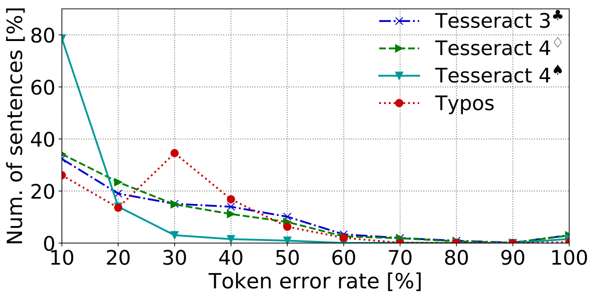
In this section, we present additional information that could facilitate reproducibility.

Hyper-parameters To train our seq2seq translation models, we generally used the default hyper-parameters of the OpenNMT toolkit. We list all non-default values in Table 8. Moreover, we decayed the learning rate eight times during the training for all models. Furthermore, we utilized *copy attention* (See et al., 2017) for our error generation models and *global attention* (Luong et al., 2015) for the error correction model.

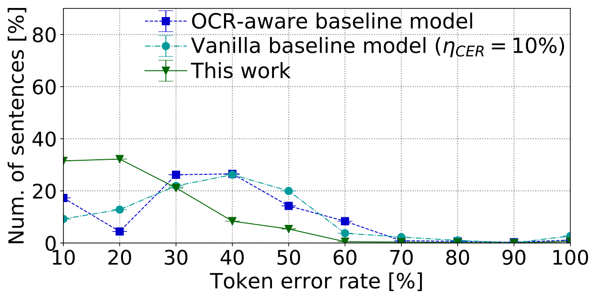
Validation Accuracy Table 9 summarizes the validation accuracy of our seq2seq models for error generation. We trained the sentence-level models for 1.6×10^4 and the token-level models for 4×10^5 iterations or at least one epoch of training. Moreover, the token-level error correction model employed by Natas was trained for one epoch (about 4×10^5 iterations) on one million parallel sentences



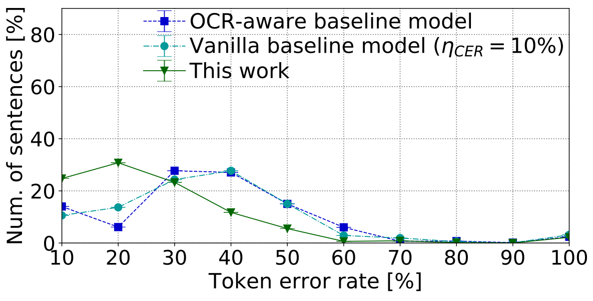
(a) Noisy sentence labeling data sets (English CoNLL 2003).



(b) Noisy sentence labeling data sets (UD English EWT).



(c) Noise induction methods (English CoNLL 2003).



(d) Noise induction methods (UD English EWT).

Figure 9: Distributions of the token error rates of sentences in our noisy sequence labeling data sets (Tesseract 3$^{\clubsuit}$, Tesseract 4$^{\diamondsuit}$, Tesseract 4$^{\spadesuit}$, and Typos). For comparison, we include error distributions obtained by applying our seq2seq token-level error generator and the baseline confusion matrix-based error models (Namysl et al., 2020) to the sentences extracted from the original benchmark. η_{CER} is the character-level noising factor used by the vanilla error model. Each point is the percentage of sentences with a token error rate that falls into a specific token error range, i.e., the value of 50 corresponds to the sentences with a token error rate greater than 40 and lower than or equal to 50.

Measure	Method	Tesser-act 3$^{\clubsuit}$	Tesser-act 4$^{\diamondsuit}$	Tesser-act 4$^{\heartsuit}$	Tesser-act 4$^{\spadesuit}$	Typos
TER	Original	22.72	16.35	14.89	3.53	15.53
	Natas	**17.24**	**12.20**	**11.13**	**2.34**	11.53
	Hunspell	17.44	13.54	12.24	2.43	**10.69**
TER (entities)	Original	29.66	16.70	15.00	3.61	8.20
	Natas	27.81	14.97	13.36	2.93	7.62
	Hunspell	**16.63**	**9.95**	**8.76**	**1.89**	**4.07**
ACC	Natas	**24.13**	**25.40**	**25.24**	**33.70**	25.75
	Hunspell	23.26	17.19	17.76	31.20	**31.17**
ACC (entities)	Natas	6.23	10.41	10.93	18.77	7.07
	Hunspell	**43.93**	**40.44**	**41.58**	**47.78**	**50.38**

(a) English CoNLL 2003.

Measure	Method	Tesser-act 3$^{\clubsuit}$	Tesser-act 4$^{\diamondsuit}$	Tesser-act 4$^{\heartsuit}$	Tesser-act 4$^{\spadesuit}$	Typos
TER	Original	23.31	22.12	20.38	5.83	15.22
	Natas	**17.76**	**17.46**	**16.23**	**4.21**	11.68
	Hunspell	19.14	19.74	18.09	4.75	**11.22**
ACC	Natas	**23.82**	**21.05**	**20.36**	**27.75**	23.27
	Hunspell	17.90	10.74	11.20	18.59	**26.49**

(b) UD English EWT.

Table 6: Token Error Rates (TER) and the correction accuracies (ACC) of Natas and Hunspell on the test sets of our noisy sequence labeling data sets. All values are percentages. **Bold** values represent the lowest TER and the highest ACC.

Noisy Token	Error-Free Token	Class Label
`No`	`No`	`O`
`nzw`	`new`	`O`
`fixtuvzs`	`fixtures`	`O`
`reported`	`reported`	`O`
`from`	`from`	`O`
`New`	`New`	`B-LOC`
`Vork`	`York`	`I-LOC`
`.`	`.`	`O`

Table 7: Example of a sentence from the noisy CoNLL 2003 data set. The first and the second column contains the noisy and the error-free tokens, respectively. The third column denotes the class label in BIO format.

and achieved 96.9% accuracy on the validation set of 5000 sentences.

Learnable Parameters The number of parameters in our sequence labeling models was constant among different models, as we used the same architecture in all experiments. The number of all model parameters was 60.3 million (including embeddings that were fixed during the training), and the number of all trainable parameters was 25.5 million. Moreover, all our seq2seq error generation and correction models had about 7.7 million parameters.

Parameter	Description	Value
`-share_vocab`	Share source and target vocabulary	True
`-share_embeddings`	Share the embeddings between encoder and decoder	True
`-word_vec_size`	Word embedding size	25
`-src_seq_length`	Max. source sequence length	1000
`-tgt_seq_length`	Max. target sequence length	1000
`-encoder_type`	Type of encoder layer	brnn
`-learning_rate`	Starting learning rate	1.0

Table 8: The hyper-parameters of the OpenNMT toolkit used to train our seq2seq error generation models.

Training set size	Validation set size	Validation accuracy	
		token-level	sentence-level
10^7	5000	98.3%	95.7%
10^6	5000	95.4%	94.9%
10^5	5000	95.1%	95.3%
10^4	1000	94.6%	90.1%
10^3	100	93.3%	91.6%

Table 9: Validation accuracy of the seq2seq models for error generation. We trained both the token-level and the sentence-level variants. The first and the second column show the number of parallel sentences used for training and validation, respectively.

Average Runtime The evaluation of the complete test set took 7 and 10 seconds on average in the case of UD English EWT and English CoNLL 2003, respectively. The runtime did not depend on the training method that was used. Nevertheless, when we employed the correction method, the runtime was significantly lengthened, e.g., it took almost 3 minutes to evaluate a model that employed the Natas correction method on English CoNLL 2003.

Computing Architecture The evaluation was performed on a workstation equipped with an Intel Xeon CPU with 10 cores and an Nvidia Quadro RTX 6000 graphics card with 24GB of memory.