The CUED's Grammatical Error Correction Systems for BEA-2019

Felix Stahlberg and **Bill Byrne**
Department of Engineering
University of Cambridge
Trumpington St, Cambridge CB2 1PZ, UK
{fs439,wjb31}@cam.ac.uk

Abstract

We describe two entries from the Cambridge University Engineering Department to the BEA 2019 Shared Task on grammatical error correction. Our submission to the low-resource track is based on prior work on using finite state transducers together with strong neural language models. Our system for the restricted track is a purely neural system consisting of neural language models and neural machine translation models trained with back-translation and a combination of checkpoint averaging and fine-tuning – without the help of any additional tools like spell checkers. The latter system has been used inside a separate system combination entry in cooperation with the Cambridge University Computer Lab.

1 Introduction

The automatic correction of errors in text [*In a such situaction* → *In such a situation*] is receiving more and more attention from the natural language processing community. A series of competitions has been devoted to grammatical error correction (GEC): the CoNLL-2013 shared task (Ng et al., 2013), the CoNLL-2014 shared task (Ng et al., 2014), and finally the BEA 2019 shared task (Bryant et al., 2019). This paper presents the contributions from the Cambridge University Engineering Department to the latest GEC competition at the BEA 2019 workshop.

We submitted systems to two different tracks. The *low-resource track* did not permit the use of parallel training data except a small development set with around 4K sentence pairs. For our low-resource system we extended our prior work on finite state transducer based GEC (Stahlberg et al., 2019) to handle new error types such as punctuation errors as well as insertions and deletions of a small number of frequent words. For the *restricted track*, the organizers provided 1.2M pairs (560K without identity mappings) of corrected and uncorrected sentences. Our goal on the restricted track was to explore the potential of purely neural models for grammatical error correction.[1] We confirm the results of Kasewa et al. (2018) and report substantial gains by applying back-translation (Sennrich et al., 2016b) to GEC – a data augmentation technique common in machine translation. Furthermore, we noticed that large parts of the training data do not match the target domain. We mitigated the domain gap by over-sampling the in-domain training corpus, and by fine-tuning through continued training. Our final model is an ensemble of four neural machine translation (NMT) models and two neural language models (LMs) with Transformer architecture (Vaswani et al., 2017). Our purely neural system was also part of the joint submission with the Cambridge University Computer Lab described by Yuan et al. (2019).

2 Low-resource Track Submission

2.1 FST-based Grammatical Error Correction

Stahlberg et al. (2019) investigated the use of finite state transducers (FSTs) for neural grammatical error correction. They proposed a cascade of FST compositions to construct a hypothesis space which is then rescored with a neural language model. We will outline this approach and explain our modifications in this section. For more details we refer to (Stahlberg et al., 2019).

In a first step, the source sentence is converted to an FST I (Fig. 1). This initial FST is augmented by composition (denoted with the $\circ$-operator) with various other FSTs to cover different error types. Composition is a widely used standard operation

[1] Models will be published at http://ucam-smt.github.io/sgnmt/html/bea19_gec.html.

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 168–175
Florence, Italy, August 2, 2019.

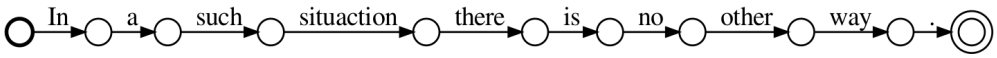

Figure 1: Input FST I representing the source sentence 'In a such situaction there is no other way.'. We follow standard convention and highlight the start state in bold and the final state with a double circle.

Figure 2: Deletion FST D which can map any token in the list R from Tab. 1 to ϵ. The σ-label matches any symbol and maps it to itself.

Deletion Frequency (dev set)	Token
164	the
78	,
50	a
33	to
20	it
18	of
16	in
12	that
8	will
8	have
8	for
8	an
7	is
7	-
6	they
6	's
6	and
5	had

Table 1: List of tokens R that can be deleted by the deletion transducer D in Fig. 2.

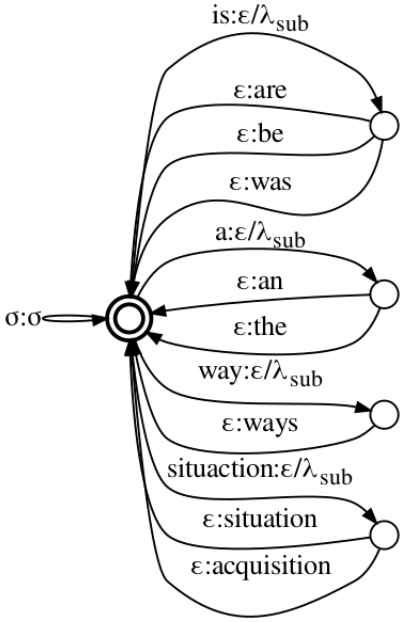

Figure 3: Edit FST E which allows substitutions with a cost of λ_{sub}. The σ-label matches any symbol and maps it to itself at no cost.

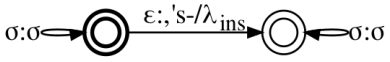

Figure 4: Insertion FST A for adding the symbols ",", "-", and "'s" at a cost of λ_{ins}. The σ-label matches any symbol and maps it to itself at no cost.

on FSTs and supported efficiently by FST toolkits such as OpenFST (Allauzen et al., 2007). We construct the hypothesis space as follows:[2]

1. We compose the input I with the deletion transducer D in Fig. 2. D allows to delete tokens on the short list shown in Tab. 1 at a cost λ_{del}. We selected R by looking up all tokens which have been deleted in the dev set more than five times and manually filtered that list slightly. We did not use the full list of dev set deletions to avoid under-estimating λ_{del} in tuning.

2. In a next step, we compose the transducer from step 1 with the edit transducer E in Fig. 3. This step addresses substitution errors such as spelling or morphology errors. Like Stahlberg et al. (2019), we use the confusion sets of Bryant and Briscoe (2018) based on CyHunspell for spell checking (Rodriguez and Seal, 2014), the AGID morphology database for morphology errors (Atkinson, 2011), and manually defined corrections for determiner and preposition errors to construct E. Additionally, we extracted all substitution errors from the BEA-2019 dev set which occurred more than five times, and added a small number of manually defined rules that fix tokenization around punctuation symbols.

3. We found it challenging to allow insertions in LM-based GEC because the LM often prefers inserting words with high unigram probability such as articles and prepositions before

[2]Note that our description differs from (Stahlberg et al., 2019) in the following ways: First, we use additional FSTs to allow insertions and deletions. Second, we integrate penalties directly into the FSTs rather than using special tokens in combination with a penalization transducer.

Sub	Del	Ins	LM	Beam	CoNLL-2014			BEA-2019 Dev		
					P	R	M2	P	R	ERRANT
Best published: Stahlberg et al. (2019)					54.12	25.52	44.21	n/a		
✓			1x	8	58.59	24.14	45.58	42.44	14.68	30.79
✓	✓		1x	8	59.01	26.07	47.11	41.21	16.47	31.69
✓	✓	✓	1x	8	52.89	26.68	44.20	40.09	19.97	33.36
✓	✓	✓	2x	8	54.05	26.71	44.87	40.70	20.01	33.73
✓	✓	✓	2x	16	57.05	27.22	46.80	42.02	19.76	34.29
✓	✓	✓	2x	32	58.48	28.21	48.15	42.37	19.92	34.58

Table 2: Results on the low-resource track. The λ-parameters are tuned on the BEA-2019 dev set.

less predictable words like proper names. We therefore restrict insertions to the three tokens ",", "-", and "'s" and allow only one insertion per sentence. We achieve this by adding the transducer A in Fig. 4 to our composition cascade.

4. Finally, we map the word-level FSTs to the subword-level by composition with a mapping transducer T that applies byte pair encoding (Sennrich et al., 2016c, BPE) to the full words. Word-to-BPE mapping transducers have been used in prior work to combine word-level models with subword-level neural sequence models (Stahlberg et al., 2019, 2017b, 2018b, 2017a).

In a more condensed form, we can describe the final transducer as:

$$I \circ D \circ E \circ A \circ T \quad (1)$$

with D for deletions, E for substitutions, A for insertions, and T for converting words to BPE tokens. Path scores in the FST in Eq. 1 are the accumulated penalties λ_{del}, λ_{sub}, and λ_{ins}. The λ-parameters are tuned on the dev set using a variant of Powell search (Powell, 1964). We apply standard FST operations like output projection, ϵ-removal, determinization, minimization, and weight pushing (Mohri, 1997; Mohri and Riley, 2001) to help downstream decoding. Following Stahlberg et al. (2019) we then use the resulting transducer to constrain a neural LM beam decoder.

2.2 Experimental Setup

Our LMs are Transformer (Vaswani et al., 2017) decoders (`transformer_big`) trained using the Tensor2Tensor library (Vaswani et al., 2018). We delay SGD updates (Stahlberg et al., 2018a; Saunders et al., 2018) with factor 2 to simulate 500K training steps with 8 GPUs on 4 physical GPUs. Training batches contain about 4K source and target tokens. Our LM training set comprises the monolingual *news2015-news2018* English training sets[3] from the WMT evaluation campaigns (Bojar et al., 2018) after language detection (Nakatani, 2010) (138M sentences) and subword segmentation using byte pair encoding (Sennrich et al., 2016c) with 32K merge operations. For decoding, we use our SGNMT tool (Stahlberg et al., 2017b, 2018b) with OpenFST backend (Allauzen et al., 2007).

2.3 Results

We report M2 (Dahlmeier and Ng, 2012) scores on the CoNLL-2014 test set (Ng et al., 2014) and span-based ERRANT scores (Bryant et al., 2017) on the BEA-2019 dev set (Bryant et al., 2019). On CoNLL-2014 we compare with the best published results with comparable amount of parallel training data. We refer to (Bryant et al., 2019) for a full comparison of BEA-2019 systems. We tune our systems on BEA-2019 and only report the performance on CoNLL-2014 for comparison to prior work.

Tab. 2 summarizes our low-resource experiments. Our substitution-only system already outperforms the prior work of Stahlberg et al. (2019). Allowing for deletions and insertions improves the ERRANT score on BEA-2019 Dev by 2.57 points. We report further gains on both test sets by ensembling two language models and increasing the beam size.

2.4 Differences Between CoNLL-2014 and BEA-2019 Dev

Our results in Tab. 2 differ significantly between the CoNLL-2014 test set and the BEA-2019 dev set. Allowing insertions is beneficial on BEA-2019 Dev but decreases the M2 score on CoNLL-2014. Increasing the beam size improves our system by 3.28 points on CoNLL-2014 while the im-

[3] http://www.statmt.org/wmt19/translation-task.html

	Per Sentence		Per Word	
	CoNLL	**BEA**	**CoNLL**	**BEA**
Missing	0.35	0.46	1.51%	2.30%
Replacement	1.52	1.31	6.62%	6.57%
Unnecessary	0.42	0.19	1.83%	0.98%
Total	2.29	1.96	9.95%	9.86%

Table 3: Number of correction types in CoNLL-2014 and BEA-2019 Dev references.

pact on BEA-2019 Dev is smaller (+0.85 points). These differences can be partially explained by comparing the error type frequencies in the reference annotations in both test sets (Tab. 3). Samples in CoNLL-2014 generally need more corrections per sentence than in BEA-2019 Dev. More importantly, the CoNLL-2014 test set contains fewer missing words, but much more unnecessary words than BEA-2019 Dev. This mismatch tempers with tuning as we explicitly tune insertion and deletion penalties.

3 Restricted Track Submission

In contrast to our low-resource submission, our restricted system entirely relies on neural models and does not use any external NLP tools, spell checkers, or hand-crafted confusion sets. For simplicity, we also chose to use standard implementations (Vaswani et al., 2018) of standard Transformer (Vaswani et al., 2017) models with standard hyper-parameters. This makes our final system easy to deploy as it is a simple ensemble of standard neural models with minimal preprocessing (subword segmentation). Our contributions on this track focus on NMT training techniques such as over-sampling, back-translation, and fine-tuning. We show that over-sampling effectively reduces domain mismatch. We found back-translation (Sennrich et al., 2016b) to be a very effective technique to utilize unannotated training data. However, while over-sampling is commonly used in machine translation to balance the number of real and back-translated training sentences, we report that using over-sampling this way for GEC hurts performance. Finally, we propose a combination of checkpoint averaging (Junczys-Dowmunt et al., 2016) and continued training to adapt our NMT models to the target domain.

3.1 Experimental Setup

We use neural LMs and neural machine translation (NMT) models in our restricted track entry.

	BASE	BIG
T2T HParams set	`trans._base`	`trans._big`
# physical GPUs	4	4
Batch size	4,192	2,048
SGD delay factor	2	4
# training iterations	300K	400K
Beam size	4	8

Table 4: NMT setups BASE and BIG used in our experiments for the restricted track.

	Number of Sentences	
	With Identities	**W/o Identities**
FCE	28K	18K
Lang-8	1,038K	498K
NUCLE	57K	21K
W&I+LOCNESS	34K	23K
Total	**1,157K**	**560K**

Table 5: BEA-2019 parallel training data with and without removing pairs where source and target sentences are the same.

Our neural LM is as described in Sec. 2.2. Our LMs and NMT models share the same subword segmentation. We perform exploratory NMT experiments with the BASE setup, but switch to the BIG setup for our final models. Tab. 4 shows the differences between both setups. Tab. 5 lists some corpus statistics for the BEA-2019 training sets. In our experiments without fine-tuning we decode with the average of the 20 most recent checkpoints (Junczys-Dowmunt et al., 2016). We use the SGNMT decoder (Stahlberg et al., 2017b, 2018b) in all our experiments.

In-domain corpus over-sampling The BEA-2019 training corpora (Tab. 5) differ significantly not only in size but also their closeness to the target domain. The W&I+LOCNESS corpus is most similar to the BEA-2019 dev and test sets in terms of domains and the distribution over English language proficiency, but only consists of 34K sentence pairs. To increase the importance of in-domain training samples we over-sampled the W&I+LOCNESS corpus with different rates. Tab. 6 shows that over-sampling by factor 4 (i.e. adding the W&I+LOCNESS corpus four times to the training set) improves the ERRAMT $F_{0.5}$-score by 2.2 points on the BEA-2019 dev set and does not lead to substantial losses on the CoNLL-2014 test set. We will over-sample the W&I+LOCNESS corpus by four in all subsequent experiments.

Removing identity mappings Previous works often suggested to remove unchanged sentences

W&I+LOCNESS Over-sampling Rate	Ratio	CoNLL-2014			BEA-2019 Dev		
		P	R	M2	P	R	ERRANT
1x	1:33	59.88	17.46	40.30	38.20	15.09	29.24
4x	1:8	59.16	17.20	39.76	40.40	16.67	31.44
8x	1:4	57.73	17.76	39.81	39.19	16.73	30.90

Table 6: Over-sampling the BEA-2019 in-domain corpus W&I+LOCNESS under BASE models. The second column contains the ratio of W&I+LOCNESS samples to training samples from the other corpora.

Identity Removal	CoNLL-2014			BEA-2019 Dev		
	P	R	M2	P	R	ERR.
×	59.16	17.20	39.76	40.40	16.67	31.44
✓	53.34	28.83	45.59	33.04	23.14	30.44

Table 7: Impact of identity removal on BASE models.

(i.e. source and target sentences are equal) from the training corpora (Stahlberg et al., 2019; Zhao et al., 2019; Grundkiewicz and Junczys-Dowmunt, 2018). We note that removing these identity mappings can be seen as measure to control the balance between precision and recall. As shown in Tab. 7, removing identities encourages the model to make more corrections and thus leads to higher recall but lower precision. It depends on the test set whether this results in an improvement in $F_{0.5}$ score. For the subsequent experiments we found that removing identities in the parallel training corpora but not in the back-translated synthetic data works well in practice.

Back-translation Back-translation (Sennrich et al., 2016b) has become the most widely used technique to use monolingual data in neural machine translation. Back-translation extends the existing parallel training set by additional training samples with real English target sentences but synthetic source sentences. Different methods have been proposed to synthesize the source sentence such as using dummy tokens (Sennrich et al., 2016b), copying the target sentence (Currey et al., 2017), or sampling from or decoding with a reverse sequence-to-sequence model (Sennrich et al., 2016b; Edunov et al., 2018; Kasewa et al., 2018). The most popular approach is to generate the synthetic source sentences with a reverse model that is trained to transform target to source sentences using beam search. In GEC, this means that the reverse model learns to introduce errors into a correct English sentence. Back-translation has been applied successfully to GEC by Kasewa et al. (2018). We confirm the effectiveness of back-translation in GEC and discuss some of the differences between applying this technique to grammatical error correction and machine translation.

Our experiments with back-translation are summarized in Tab. 8. Adding 1M synthetic sentences to the training data already yields very substantial gains on both test sets. We achieve our best results with 5M synthetic sentences (+8.44 on BEA-2019 Dev). In machine translation, it is important to maintain a balance between authentic and synthetic data (Sennrich et al., 2016b; Poncelas et al., 2018; Sennrich et al., 2016a). Over-sampling the real data is a common practice to rectify that ratio if large amounts of synthetic data are available. Interestingly, over-sampling real data in GEC hurts performance (row 3 vs. 5 in Tab. 8), and it is possible to mix real and synthetic sentences at a ratio of 1:7.9 (last three rows in Tab. 8). We will proceed with the 5M setup for the remainder of this paper.

Fine-tuning As explained previously, we over-sample the W&I+LOCNESS corpus by factor 4 to mitigate the domain gap between the training set and the BEA-2019 dev and test sets. To further adapt our system to the target domain, we fine-

Over-sampling Rate (Real Data)	Number of Synthetic Sentences	Ratio	CoNLL-2014			BEA-2019 Dev		
			P	R	M2	P	R	ERRANT
1x	0	-	53.34	28.83	45.59	33.04	23.14	30.44
1x	1M	1:1.6	56.17	31.30	48.47	37.79	23.86	33.84
1x	3M	1:4.8	61.40	34.29	53.02	42.62	25.30	37.49
1x	5M	1:7.9	64.18	34.27	54.64	44.69	25.59	38.88
3x	3M	1:1.6	57.12	32.55	49.63	40.08	24.79	35.68
6x	5M	1:1.3	59.15	33.99	51.52	41.52	25.05	36.69

Table 8: Using back-translation for GEC (BASE models). The third column contains the ratio between real and synthetic sentence pairs.

Fine-tuning (Continued Training)	Checkpoint Averaging	CoNLL-2014			BEA-2019 Dev		
		P	R	M2	P	R	ERRANT
		63.61	33.39	53.86	44.16	25.01	38.29
	✓	64.18	34.27	54.64	44.69	25.59	38.88
✓		64.98	33.05	54.46	48.62	27.19	42.00
✓	✓	66.03	34.17	55.65	48.99	26.87	42.06

Table 9: Fine-tuning through continued training on W&I+LOCNESS and checkpoint averaging with a BASE model with 5M back-translated sentences.

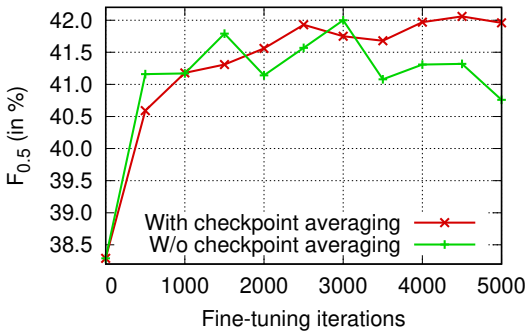

Figure 5: Span-based ERRANT $F_{0.5}$ scores on the BEA-2019 dev set over the number of fine-tuning training iterations (single GPU, SGD delay factor (Saunders et al., 2018) of 16).

tune the NMT models on W&I+LOCNESS after convergence on the full training set. We do that by continuing training on W&I+LOCNESS from the last checkpoint of the first training pass. Fig. 5 plots the $F_{0.5}$ score on the BEA-2019 dev set for two different setups. For the red curve, we average all checkpoints (Junczys-Dowmunt et al., 2016) (including the last unadapted checkpoint) up to a certain training iteration. Checkpoints are dumped every 500 steps. The green curve does not use any checkpoint averaging. Checkpoint averaging helps to smooth out fluctuations in $F_{0.5}$ score, and also generalizes better to CoNLL-2014 (Tab. 9).

Final system Tab. 10 contains our experiments with the BIG configuration. In addition to W&I+LOCNESS over-sampling, back-translation with 5M sentences, and fine-tuning with checkpoint averaging, we report further gains by adding the language models from our low-resource system (Sec. 2.2) and ensembling. Our best system (4 NMT models, 2 language models) achieves 58.9 M2 on CoNLL-2014, which is slightly (2.25 points) worse than the best published result on that test set (Zhao et al., 2019). However, we note that we have tailored our system towards the BEA-2019 dev set and not the CoNLL-2013 or CoNLL-2014 test sets. As we argued in Sec. 2.4, our results throughout this work suggest strongly that the optimal system parameters for these test sets are very different from each other, and that our final system settings are not optimal for CoNLL-2014. We also note that unlike the system of Zhao et al. (2019), our system for the restricted track does not use spell checkers or other NLP tools but relies solely on neural sequence models.

4 Conclusion

We participated in the BEA 2019 Shared Task on grammatical error correction with submissions to the low-resource and the restricted track. Our low-resource system is an extension of prior work on FST-based GEC (Stahlberg et al., 2019) to allow insertions and deletions. Our restricted track submission is a purely neural system based on standard NMT and LM architectures. We pointed out the similarity between GEC and machine translation, and demonstrated that several techniques which originate from MT research such as over-sampling, back-translation, and fine-tuning, are also useful for GEC. Our models have been used in a joint submission with the Cambridge University Computer Lab (Yuan et al., 2019).

NMT	Fine-tuning	LM	CoNLL-2014			BEA-2019 Dev		
			P	R	M2	P	R	ERRANT
Best published: Zhao et al. (2019)			71.57	38.65	61.15	n/a		
1x			64.04	35.74	55.28	45.86	26.46	40.00
1x	✓		66.57	35.21	56.50	51.57	27.49	43.88
1x	✓	2x	61.53	40.44	55.72	48.30	33.08	44.23
4x	✓		70.37	35.12	58.60	55.84	27.80	46.47
4x	✓	2x	66.89	39.85	58.90	53.17	32.89	47.34

Table 10: Final results on the restricted track with BIG models and back-translation.

Acknowledgments

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC) grant EP/L027623/1 and has been performed using resources provided by the Cambridge Tier-2 system operated by the University of Cambridge Research Computing Service[4] funded by EPSRC Tier-2 capital grant EP/P020259/1.

References

Cyril Allauzen, Michael Riley, Johan Schalkwyk, Wojciech Skut, and Mehryar Mohri. 2007. OpenFst: A general and efficient weighted finite-state transducer library. In *Implementation and Application of Automata*, pages 11–23. Springer.

Kevin Atkinson. 2011. Automatically generated inflection database (AGID). http://wordlist.aspell.net/other/. [Online; accessed 24 April 2019].

Ondřej Bojar, Christian Federmann, Mark Fishel, Yvette Graham, Barry Haddow, Philipp Koehn, and Christof Monz. 2018. Findings of the 2018 conference on machine translation (WMT18). In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 272–303, Belgium, Brussels. Association for Computational Linguistics.

Christopher Bryant and Ted Briscoe. 2018. Language model based grammatical error correction without annotated training data. In *Proceedings of the Thirteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 247–253. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.

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

Anna Currey, Antonio Valerio Miceli Barone, and Kenneth Heafield. 2017. Copied monolingual data improves low-resource neural machine translation. In *Proceedings of the Second Conference on Machine Translation*, pages 148–156, Copenhagen, Denmark. Association for Computational Linguistics.

[4] http://www.hpc.cam.ac.uk

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572. 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.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Tomasz Dwojak, and Hieu Hoang. 2016. Is neural machine translation ready for deployment? A case study on 30 translation directions. In *International Workshop on Spoken Language Translation IWSLT*.

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.

Mehryar Mohri. 1997. Finite-state transducers in language and speech processing. *Computational Linguistics*, 23(2).

Mehryar Mohri and Michael Riley. 2001. A weight pushing algorithm for large vocabulary speech recognition. In *Seventh European Conference on Speech Communication and Technology*.

Shuyo Nakatani. 2010. Language detection library for Java. [Online; accessed 24 April 2019].

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. 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. Association for Computational Linguistics.

Alberto Poncelas, Dimitar Shterionov, Andy Way, Gideon Maillette de Buy Wenniger, and Peyman Passban. 2018. Investigating backtranslation in neural machine translation. *arXiv preprint arXiv:1804.06189*.

Michael JD Powell. 1964. An efficient method for finding the minimum of a function of several variables without calculating derivatives. *The computer journal*, 7(2):155–162.

Tim Rodriguez and Matthew Seal. 2014. Cy-Hunspell. `https://github.com/MSeal/cython_hunspell`. [Online; accessed 24 April 2019].

Danielle Saunders, Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018. Multi-representation ensembles and delayed SGD updates improve syntax-based NMT. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 319–325, Melbourne, Australia. Association for Computational Linguistics.

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

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

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016c. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Felix Stahlberg, Christopher Bryant, and Bill Byrne. 2019. Neural grammatical error correction with finite state transducers. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.

Felix Stahlberg, Adrià de Gispert, and Bill Byrne. 2018a. The University of Cambridge's machine translation systems for WMT18. In *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 504–512, Belgium, Brussels. Association for Computational Linguistics.

Felix Stahlberg, Adrià de Gispert, Eva Hasler, and Bill Byrne. 2017a. Neural machine translation by minimising the Bayes-risk with respect to syntactic translation lattices. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 362–368, Valencia, Spain. Association for Computational Linguistics.

Felix Stahlberg, Eva Hasler, Danielle Saunders, and Bill Byrne. 2017b. SGNMT – A flexible NMT decoding platform for quick prototyping of new models and search strategies. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 25–30. Association for Computational Linguistics.

Felix Stahlberg, Danielle Saunders, Gonzalo Iglesias, and Bill Byrne. 2018b. Why not be versatile? Applications of the SGNMT decoder for machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 208–216. Association for Machine Translation in the Americas.

Ashish Vaswani, Samy Bengio, Eugene Brevdo, Francois Chollet, Aidan Gomez, Stephan Gouws, Llion Jones, Łukasz Kaiser, Nal Kalchbrenner, Niki Parmar, Ryan Sepassi, Noam Shazeer, and Jakob Uszkoreit. 2018. Tensor2tensor for neural machine translation. In *Proceedings of the 13th Conference of the Association for Machine Translation in the Americas (Volume 1: Research Papers)*, pages 193–199, Boston, MA. Association for Machine Translation in the Americas.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, pages 5998–6008.

Zheng Yuan, Felix Stahlberg, Marek Rei, Bill Byrne, and Helen Yannakoudakis. 2019. Neural and FST-based approaches to grammatical error correction. In *Proceedings of the 14th workshop on innovative use of NLP for building educational applications*. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*. Association for Computational Linguistics.