Neural Grammatical Error Correction with Finite State Transducers

Felix Stahlberg[†] and **Christopher Bryant**[‡] and **Bill Byrne**[†]
[†]Department of Engineering
[‡] Department of Computer Science and Technology
University of Cambridge
United Kingdom
{fs439, cjb255, wjb31}@cam.ac.uk

Abstract

Grammatical error correction (GEC) is one of the areas in natural language processing in which purely neural models have not yet superseded more traditional symbolic models. Hybrid systems combining phrase-based statistical machine translation (SMT) and neural sequence models are currently among the most effective approaches to GEC. However, both SMT and neural sequence-to-sequence models require large amounts of annotated data. Language model based GEC (LM-GEC) is a promising alternative which does not rely on annotated training data. We show how to improve LM-GEC by applying modelling techniques based on finite state transducers. We report further gains by rescoring with neural language models. We show that our methods developed for LM-GEC can also be used with SMT systems if annotated training data is available. Our best system outperforms the best published result on the CoNLL-2014 test set, and achieves far better relative improvements over the SMT baselines than previous hybrid systems.

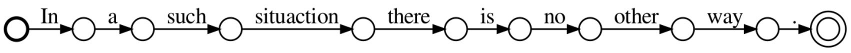
1 Introduction

Grammatical error correction (GEC) is the task of automatically correcting all types of errors in text; e.g. [*In a such situation* → *In such a situation*]. Using neural models for GEC is becoming increasingly popular (Xie et al., 2016; Yuan and Briscoe, 2016; Ji et al., 2017; Sakaguchi et al., 2017; Schmaltz et al., 2017; Chollampatt and Ng, 2018; Ge et al., 2018a,b), possibly combined with phrase-based SMT (Chollampatt et al., 2016; Chollampatt and Ng, 2017; Grundkiewicz and Junczys-Dowmunt, 2018). A potential challenge for purely neural GEC models is their vast output space since they assign non-zero probability mass to any sequence. GEC is – compared to machine translation – a highly constrained problem as corrections tend to be very local, and lexical choices are usually limited. Finite state transducers (FSTs) are an efficient way to represent large structured search spaces. In this paper, we propose to construct a hypothesis space using standard FST operations like composition, and then constrain the output of a neural GEC system to that space. We study two different scenarios: In the first scenario, we do not have access to annotated training data, and only use a small development set for tuning. In this scenario, we construct the hypothesis space using word-level context-independent confusion sets (Bryant and Briscoe, 2018) based on spell checkers and morphology databases, and rescore it with count-based and neural language models (NLMs). In the second scenario, we assume to have enough training data available to train SMT and neural machine translation (NMT) systems. In this case, we make additional use of the SMT lattice and rescore with an NLM-NMT ensemble. Our contributions are:

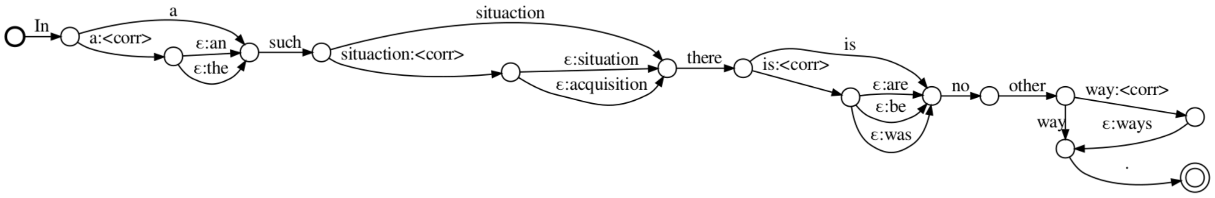
- We present an FST-based adaptation of the work of Bryant and Briscoe (2018) which allows exact inference, and does not require annotated training data. We report large gains from rescoring with a neural language model.
- Our technique beats the best published result with comparable amounts of training data on the CoNLL-2014 (Ng et al., 2014) test set when applied to SMT lattices. Our combination strategy yields larger gains over the SMT baselines than simpler rescoring or pipelining used in prior work on hybrid systems (Grundkiewicz and Junczys-Dowmunt, 2018).

2 Constructing the Hypothesis Space

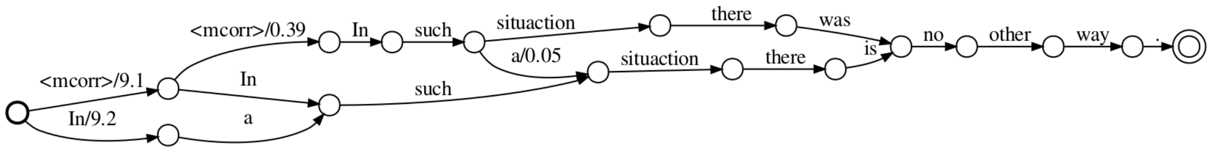
Constructing the set of hypotheses The core idea of our approach is to first construct a



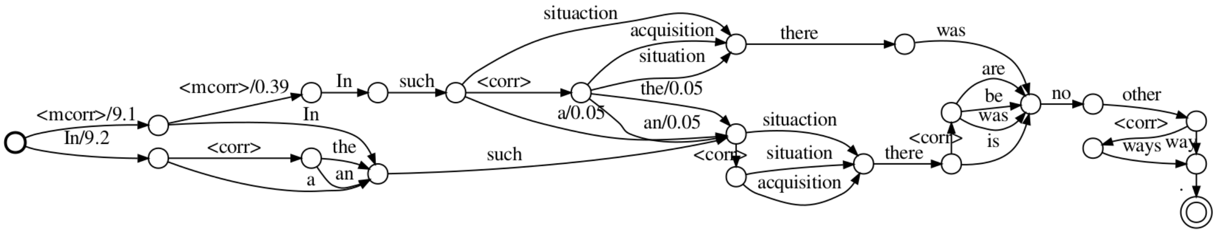
(a) The input lattice I without SMT (no annotated training data).



(b) The base lattice B without SMT.



(c) The input lattice I with SMT.



(d) The base lattice B with SMT.

Figure 1: Building the hypothesis space for the input sentence "In a such situaction there is no other way .".

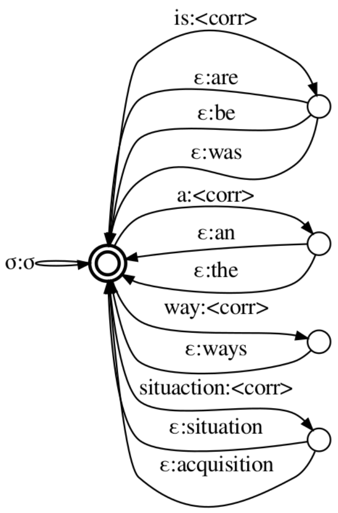


Figure 2: The edit transducer E. The σ-label can match any input symbol.

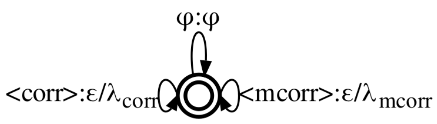


Figure 3: The penalization transducer P. The ϕ-label can match any input except <corr> and <mcorr>.

(weighted) hypothesis space H which is large enough to be likely to contain good corrections, but constrained enough to embrace the highly structured nature of GEC. Then, we use H to constrain a neural beam decoder. We make extensive use of the FST operations available in OpenFST (Allauzen et al., 2007) like composition (denoted with the $\circ$-operator) and projection (denoted with $\Pi_{\text{input}}(\cdot)$ and $\Pi_{\text{output}}(\cdot)$) to build H. The process starts with an input lattice I. In our experiments without annotated training data, I is an FST which simply maps the input sentence to itself as shown in Fig. 1(a). If we do have access to enough annotated data, we train an SMT system on it and derive I from the SMT n-best list.[1] For each hypothesis $\mathbf{y}$ we compute the Levenshtein distance $\text{lev}(\mathbf{x}, \mathbf{y})$ to the source sentence $\mathbf{x}$. We construct a string $\mathbf{z}$ by prepending $\text{lev}(\mathbf{x}, \mathbf{y})$ many <mcorr> tokens to $\mathbf{y}$, and construct I such that:

$$\mathbf{z} = (\texttt{<mcorr>})^{\text{lev}(\mathbf{x},\mathbf{y})} \cdot \mathbf{y} \tag{1}$$

$$[[I]](\mathbf{z}) = -\lambda_{\text{SMT}} \text{SMT}(\mathbf{y}|\mathbf{x}). \tag{2}$$

[1]In the rare cases in which the n-best list did not contain the source sentence $\mathbf{x}$ we added it in a postprocessing step.

We adapt the notation of Mohri (2003) and denote the cost I assigns to mapping a string $\mathbf{z}$ to itself as $[[I]](\mathbf{z})$, and set $[[I]](\mathbf{z}) = \infty$ if I does not accept $\mathbf{z}$. $\text{SMT}(\mathbf{y}|\mathbf{x})$ is the SMT score. In other words, I represents the weighted SMT n-best list after adding $\text{lev}(\mathbf{x}, \mathbf{y})$ many `<mcorr>` tokens to each hypothesis as illustrated in Fig. 1(c). We scale SMT scores by a factor λ_{SMT} for tuning.

Bryant and Briscoe (2018) addressed substitution errors such as non-words, morphology-, article-, and preposition-errors by creating confusion sets $C(x_i)$ that contain possible (context-independent) 1:1 corrections for each input word x_i. Specifically, they relied on CyHunspell for spell checking (Rodriguez and Seal, 2014), the AGID morphology database for morphology errors (Atkinson, 2011), and manually defined confusion sets for determiner and preposition errors, hence avoiding the need for annotated training data. We use the same confusion sets as Bryant and Briscoe (2018) to augment our hypothesis space via the edit flower transducer E shown in Fig. 2. E can map any sequence to itself via its σ-self-loop. Additionally, it allows the mapping $x_i \rightarrow \texttt{<corr>} \cdot y$ for each $y \in C(x_i)$. For example, for the misspelled word $x_i =$ 'situaction' and the confusion set $C(\text{'situaction'}) = \{\text{'situation'}, \text{'acquisition'}\}$, E allows mapping 'situaction' to '`<corr>` situation' and '`<corr>` acquisition', and to itself via the σ-self-loop. The additional `<corr>` token will help us to keep track of the edits. We obtain our *base lattice* B which defines the set of possible hypotheses by composition and projection:

$$B := \Pi_{\text{output}}(I \circ E). \tag{3}$$

Fig. 1(d) shows B for our running example.

Scoring the hypothesis space We apply multiple scoring strategies to the hypotheses in B. First, we penalize `<mcorr>` and `<corr>` tokens with two further parameters, λ_{mcorr} and λ_{corr}, by composing B with the penalization transducer P shown in Fig. 3.[2] The λ_{mcorr} and λ_{corr} parameters control the trade-off between the number and quality of the proposed corrections since high values bias towards fewer corrections.

To incorporate word-level language model scores we train a 5-gram count-based LM with KenLM (Heafield, 2011) on the One Billion Word Benchmark dataset (Chelba et al., 2014), and convert it to an FST L using the OpenGrm NGram Library (Roark et al., 2012). For tuning purposes we scale weights in L with λ_{KenLM}:

$$[[L]](\mathbf{y}) = -\lambda_{\text{KenLM}} \log P_{\text{KenLM}}(\mathbf{y}). \tag{4}$$

Our combined word-level scores can be expressed with the following transducer:

$$H_{\text{word}} = B \circ P \circ L. \tag{5}$$

Since we operate in the tropical semiring, path scores in H_{word} are linear combinations of correction penalties, LM scores, and, if applicable, SMT scores, weighted with the λ-parameters. Note that exact inference in H_{word} is possible using FST shortest path search. This is an improvement over the work of Bryant and Briscoe (2018) who selected correction options greedily. Our ultimate goal, however, is to rescore H_{word} with neural models such as an NLM and – if annotated training data is available – an NMT model. Since our neural models use subword units (Sennrich et al., 2016, BPEs), we compose H_{word} with a transducer T which maps word sequences to BPE sequences. Our final transducer H_{BPE} which we use to constrain the neural beam decoder can be written as:

$$\begin{aligned} H_{\text{BPE}} &= \Pi_{\text{output}}(H_{\text{word}} \circ T) \\ &= \Pi_{\text{output}}(I \circ E \circ P \circ L \circ T). \end{aligned} \tag{6}$$

To help downstream beam decoding we apply ϵ-removal, determinization, minimization, and weight pushing (Mohri, 1997; Mohri and Riley, 2001) to H_{BPE}. We search for the best hypothesis $\mathbf{y}^*_{\text{BPE}}$ with beam search using a combined score of word-level symbolic models (represented by H_{BPE}) and subword unit based neural models:

$$\begin{aligned} \mathbf{y}^*_{\text{BPE}} = \arg\max_{\mathbf{y}_{\text{BPE}}} \Big(&- [[H_{\text{BPE}}]](\mathbf{y}_{\text{BPE}}) \\ &+ \lambda_{\text{NLM}} \log P_{\text{NLM}}(\mathbf{y}_{\text{BPE}}) \\ &+ \lambda_{\text{NMT}} \log P_{\text{NMT}}(\mathbf{y}_{\text{BPE}}|\mathbf{x}_{\text{BPE}}) \Big) \end{aligned} \tag{7}$$

The final decoding pass can be seen as an ensemble of a neural LM and an NMT model which is constrained and scored at each time step by the set of possible tokens in H_{BPE}.

[2]Rather than using `<mcorr>` and `<corr>` tokens and the transducer P we could directly incorporate the costs in the transducers I and E, respectively. We chose to use explicit correction tokens for clarity.

	Uses E	5-gram FST-LM	NLM (BPE)	CoNLL-2014 P	R	M2	GLEU	JFLEG Test P	R	M2	GLEU
1	Best published (B&B, 2018)			40.56	20.81	34.09	59.35	76.23	28.48	57.08	48.75
2	✓	✓		40.62	20.72	34.08	64.03	81.08	28.69	59.38	48.95
3	✓	✓	✓	54.43	25.21	44.19	66.75	79.88	32.99	62.20	50.93
4	✓	✓	✓	53.64	26.34	44.43	66.89	70.24	38.94	60.51	52.61

Table 1: Results without using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the source sentence as in Fig. 1(a).

	Uses E	5-gram FST-LM	NMT (BPE)	NLM (BPE)	CoNLL-2014 P	R	M2	GLEU	JFLEG Test P	R	M2	GLEU
1	Best published (G&J-D, 2018)				66.77	34.49	56.25	n/a	n/a	n/a	n/a	61.50
2	Unconstrained single NMT				54.98	22.20	42.45	67.19	67.49	38.47	58.64	50.71
3					60.95	26.21	48.18	68.30	66.64	40.68	59.09	50.86
4	✓	✓			57.58	32.39	49.83	68.82	71.60	42.45	62.95	53.20
5			✓	✓	65.26	33.03	54.61	69.92	76.35	40.55	64.89	51.75
6	✓		✓	✓	64.55	37.33	56.33	70.30	78.85	47.72	69.75	55.39
7	✓		✓(4x)	✓	66.71	38.97	58.40	70.60	82.15	47.82	71.84	55.60
8	✓		✓(4x)	✓	66.96	38.62	58.39	70.60	74.19	56.41	69.79	58.63

Table 2: Results with using annotated training data. Systems are tuned with respect to the metric highlighted in gray. Input lattices I are derived from the Moses 1000-best list as in Fig. 1(c). Row 3 is the SMT baseline.

We have introduced three λ-parameters λ_{corr}, λ_{KenLM}, and λ_{NLM}, and three additional parameters λ_{SMT}, λ_{mcorr}, and λ_{NMT} if we make use of annotated training data. We also use a word insertion penalty λ_{wc} for our SMT-based experiments. We tune all these parameters on the development sets using Powell search (Powell, 1964).[3]

3 Experiments

Experimental setup In our experiments with annotated training data we use the SMT system of Junczys-Dowmunt and Grundkiewicz (2016)[4] to create 1000-best lists from which we derive the input lattices I. All our LMs are trained on the One Billion Word Benchmark dataset (Chelba et al., 2014). Our neural LM is a Transformer decoder architecture in the `transformer_base` configuration trained with Tensor2Tensor (Vaswani et al., 2018). Our NMT model is a Transformer model (`transformer_base`) trained on the concatenation of the NUCLE corpus (Dahlmeier et al., 2013) and the Lang-8 Corpus of Learner English v1.0 (Mizumoto et al., 2012). We only keep sentences with at least one correction (659K sentences in total). Both NMT and NLM models use byte pair encoding (Sennrich et al., 2016, BPE) with 32K merge operations. We delay SGD updates by 2 on four physical GPUs as suggested by Saunders et al. (2018). We decode with beam size 12 using the SGNMT decoder (Stahlberg et al., 2017). We evaluate on CoNLL-2014 (Ng et al., 2014) and JFLEG-Test (Napoles et al., 2017), using CoNLL-2013 (Ng et al., 2013) and JFLEG-Dev as development sets. Our evaluation metrics are GLEU (Napoles et al., 2015) and M2 (Dahlmeier and Ng, 2012). We generated M2 files using ERRANT (Bryant et al., 2017) for JFLEG and Tab. 1 to be comparable to Bryant and Briscoe (2018), but used the official M2 files in Tab. 2 to be comparable to Grundkiewicz and Junczys-Dowmunt (2018).

Results Our LM-based GEC results without using annotated training data are summarized in Tab. 1. Even when we use the same resources (same LM and same confusion sets) as Bryant and Briscoe (2018), we see gains on JFLEG (rows 1 vs. 2), probably because we avoid search errors in our FST-based scheme. Adding an NLM yields significant gains across the board. Tab. 2 shows that adding confusion sets to SMT lattices is effective even without neural models (rows 3 vs. 4). Rescoring with neural models also benefits from the confusion sets (rows 5 vs. 6). With our ensemble systems (rows 7 and 8) we are able to outperform prior work[5] (row 1) on CoNLL-2014 and

[3] Similarly to Bryant and Briscoe (2018), even in our experiments without annotated *training* data, we do need a very small amount of annotated sentences for tuning.

[4] `https://github.com/grammatical/baselines-emnlp2016`

[5] We compare our systems to the work of Grundkiewicz and Junczys-Dowmunt (2018) as they used similar training data. We note, however, that Ge et al. (2018b) reported even better results with much more (non-public) training data. Comparing (Ge et al., 2018a) and (Ge et al., 2018b) suggests that most of their gains come from the larger training set.

	G&J-D (2018)		This work	
	CoNLL (M2)	JFLEG (GLEU)	CoNLL (M2)	JFLEG (GLEU)
SMT	50.27	55.79	48.18	50.86
Hybrid	56.25	61.50	58.40	58.63
Rel. gain	**11.90%**	**10.23%**	**21.21%**	**15.28%**

Table 3: Improvements over SMT baselines.

come within 3 GLEU on JFLEG. Since the baseline SMT systems of Grundkiewicz and Junczys-Dowmunt (2018) were better than the ones we used, we achieve even higher relative gains over the respective SMT baselines (Tab. 3).

Error type analysis We also carried out a more detailed error type analysis of the best CoNLL-2014 M2 system with/without training data using ERRANT (Tab. 4). Specifically, this table shows that while the trained system was consistently better than the untrained system, the degree of the improvement differs significantly depending on the error type. In particular, since the untrained system was only designed to handle Replacement word errors, much of the improvement in the trained system comes from the ability to correct Missing and Unnecessary word errors. The trained system nevertheless still improves upon the untrained system in terms of replacement errors by 10 $F_{0.5}$ (45.53 vs. 55.63).

In terms of more specific error types, the trained system was also able to capture a wider variety of error types, including content word errors (adjectives, adverbs, nouns and verbs) and other categories such as pronouns and punctuation. Since the untrained system only targets spelling, orthographic and morphological errors however, it is interesting to note that the difference in scores between these categories tends to be smaller than others; e.g. noun number (53.43 vs 64.96), orthography (62.77 vs 74.07), spelling (67.91 vs 75.21) and subject-verb agreement (66.67 vs 68.39). This suggests that an untrained system is already able to capture the majority of these error types.

Oracle experiments Our FST-based composition cascade is designed to enrich the search space to allow the neural models to find better hypotheses. Tab. 5 reports the oracle sentence error rate for different configurations, i.e. the fraction of reference sentences in the test set which are not in the FSTs. Expanding the SMT lattice significantly reduces the oracle error rate from 55.63% to 48.17%.

	ERRANT $F_{0.5}$	
Type	**No train**	**Train**
Missing	-	51.96
Replacement	45.53	55.63
Unnecessary	-	50.38
ADJ	-	27.03
ADV	-	29.80
DET	19.17	55.01
MORPH	33.20	64.81
NOUN	4.31	34.88
NOUN:NUM	53.43	64.96
NOUN:POSS	-	13.51
ORTH	62.77	74.07
OTHER	2.45	18.39
PREP	34.39	56.58
PRON	-	40.91
PUNCT	-	46.08
SPELL	67.91	75.21
VERB	-	37.94
VERB:FORM	48.03	63.33
VERB:SVA	66.67	68.39
VERB:TENSE	35.39	47.90

Table 4: A selection of ERRANT $F_{0.5}$ error type scores comparing the best CoNLL-2014 system with and without training data. A dash means the system did not attempt to correct the error type.

Hypothesis space	**Error rate**
Expanded input sentence (Tab. 1)	61.28%
SMT lattice (Tab. 2, rows 3, 5)	55.64%
Expanded SMT lattice (Tab. 2, rows 4, 6-8)	48.17%

Table 5: Oracle sentence error rates for different hypothesis spaces using the first annotator in CoNLL-2014.

4 Conclusion

We demonstrated that our FST-based approach to GEC outperforms prior work on LM-based GEC significantly, especially when combined with a neural LM. We also applied our approach to SMT lattices and reported much better relative gains over the SMT baselines than previous work on hybrid systems. Our results suggest that FSTs provide a powerful and effective framework for constraining neural GEC systems.

Acknowledgements

This work was supported by the U.K. Engineering and Physical Sciences Research Council (EPSRC grant EP/L027623/1).

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-December-2018].

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, 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.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2014. One billion word benchmark for measuring progress in statistical language modeling. In *Fifteenth Annual Conference of the International Speech Communication Association (INTERSPEECH-2014)*, pages 2635–2639.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. In *Proceedings of the 12th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 327–333. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, New Orleans, Louisiana, USA.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. In *Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence*, pages 2768–2774. AAAI Press.

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.

Daniel Dahlmeier, Hwee Tou Ng, and Siew Mei Wu. 2013. Building a large annotated corpus of learner English: The NUS corpus of learner English. In *Proceedings of the Eighth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 22–31. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018a. Fluency boost learning and inference for neural grammatical error correction. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1055–1065. Association for Computational Linguistics.

Tao Ge, Furu Wei, and Ming Zhou. 2018b. Reaching human-level performance in automatic grammatical error correction: An empirical study. *arXiv preprint arXiv:1807.01270*.

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.

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

Jianshu Ji, Qinlong Wang, Kristina Toutanova, Yongen Gong, Steven Truong, and Jianfeng Gao. 2017. A nested attention neural hybrid model for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 753–762. Association for Computational Linguistics.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1546–1556. Association for Computational Linguistics.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of ESL writings. In *Proceedings of COLING 2012: Posters*, pages 863–872. The COLING 2012 Organizing Committee.

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

Mehryar Mohri. 2003. Edit-distance of weighted automata: General definitions and algorithms. *International Journal of Foundations of Computer Science*, 14(06):957–982.

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

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

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

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.

Brian Roark, Richard Sproat, Cyril Allauzen, Michael Riley, Jeffrey Sorensen, and Terry Tai. 2012. The OpenGrm open-source finite-state grammar software libraries. In *Proceedings of the ACL 2012 System Demonstrations*, pages 61–66. Association for Computational Linguistics.

Tim Rodriguez and Matthew Seal. 2014. CyHunspell. `https://github.com/MSeal/cython_hunspell`. [Online; accessed 24-December-2018].

Keisuke Sakaguchi, Matt Post, and Benjamin Van Durme. 2017. Grammatical error correction with neural reinforcement learning. In *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 366–372. Asian Federation of Natural Language Processing.

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. Association for Computational Linguistics.

Allen Schmaltz, Yoon Kim, Alexander Rush, and Stuart Shieber. 2017. Adapting sequence models for sentence correction. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 2807–2813. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. 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, Eva Hasler, Danielle Saunders, and Bill Byrne. 2017. 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.

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. Association for Machine Translation in the Americas.

Ziang Xie, Anand Avati, Naveen Arivazhagan, Dan Jurafsky, and Andrew Y Ng. 2016. Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

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. Association for Computational Linguistics.