How do you correct run-on sentences it's not as easy as it seems

Junchao Zheng, Courtney Napoles, Joel Tetreault and **Kostiantyn Omelianchuk**

Grammarly

`firstname.lastname@grammarly.com`

Abstract

Run-on sentences are common grammatical mistakes but little research has tackled this problem to date. This work introduces two machine learning models to correct run-on sentences that outperform leading methods for related tasks, punctuation restoration and whole-sentence grammatical error correction. Due to the limited annotated data for this error, we experiment with artificially generating training data from clean newswire text. Our findings suggest artificial training data is viable for this task. We discuss implications for correcting run-ons and other types of mistakes that have low coverage in error-annotated corpora.

1 Introduction

A run-on sentence is defined as having at least two main or independent clauses that lack either a conjunction to connect them or a punctuation mark to separate them. Run-ons are problematic because they not only make the sentence unfriendly to the reader but potentially also to the local discourse. Consider the example in Table 1.

In the field of grammatical error correction (GEC), most work has typically focused on determiner, preposition, verb and other errors which non-native writers make more frequently. Run-ons have received little to no attention even though they are common errors for both native and non-native speakers. Among college students in the United States, run-on sentences are the 18th most frequent error and the 8th most frequent error made by students who are not native English speakers (Leacock et al., 2014).

Correcting run-on sentences is challenging (Kagan, 1980) for several reasons:

- They are sentence-level mistakes with long-distance dependencies, whereas most other grammatical errors are local and only need a small window for decent accuracy.

Before correction
But the illiterate will not stay illiterate always if they put an effort to improve and are given a chance for good education, they can still develop into a group of productive Singaporeans.

After correction
But the illiterate will not stay illiterate always. If they put an effort to improve and are given a chance for good education, they can still develop into a group of productive Singaporeans.

Table 1: A run-on sentence before and after correction.

- There are multiple ways to fix a run-on sentence. For example, one can a) add sentence-ending punctuation to separate them; b) add a conjunction (such as *and*) to connect the two clauses; or c) convert an independent clause into a dependent clause.

- They are relatively infrequent in existing, annotated GEC corpora and therefore existing systems tend not to learn how to correct them.

In this paper, we analyze the task of automatically correcting run-on sentences. We develop two methods: a conditional random field model (roCRF) and a Seq2Seq attention model (roS2S) and show that they outperform models from the sister tasks of punctuation restoration and whole-sentence grammatical error correction. We also experiment with artificially generating training examples in clean, otherwise grammatical text, and show that models trained on this data do nearly as well predicting artificial and naturally occurring run-on sentences.

2 Related Work

Early work in the field of GEC focused on correcting specific error types such as preposition and article errors (Tetreault et al., 2010; Rozovskaya and Roth, 2011; Dahlmeier and Ng, 2011), but did not consider run-on sentences. The closest work to our own is Israel et al. (2012), who used Conditional Random Fields (CRFs) for correcting comma errors (excluding comma splices, a type of run-on sentence). Lee et al. (2014) used a similar system based on CRFs but focused on comma splice correction. Recently, the field has focused on the task of *whole-sentence correction*, targeting all errors in a sentence in one pass. Whole-sentence correction methods borrow from advances in statistical machine translation (Madnani et al., 2012; Felice et al., 2014; Junczys-Dowmunt and Grundkiewicz, 2016) and, more recently, neural machine translation (Yuan and Briscoe, 2016; Chollampatt and Ng, 2018; Xie et al., 2018; Junczys-Dowmunt et al., 2018).

To date, GEC systems have been evaluated on corpora of non-native student writing such as NUCLE (Dahlmeier et al., 2013) and the Cambridge Learner Corpus First Certificate of English (Yannakoudakis et al., 2011). The 2013 and 2014 CoNLL Shared Tasks in GEC used NUCLE as their train and test sets (Ng et al., 2013, 2014). There are few instances of run-on sentences annotated in both test sets, making it hard to assess system performance on that error type.

A closely related task to run-on error correction is that of *punctuation restoration* in the automatic speech recognition (ASR) field. Here, a system takes as input a speech transcription and is tasked with inserting any type of punctuation where appropriate. Most work utilizes textual features with n-gram models (Gravano et al., 2009), CRFs (Lu and Ng, 2010), convolutional neural networks or recurrent neural networks (Peitz et al., 2011; Che et al., 2016). The Punctuator (Tilk and Alumäe, 2016) is a leading punctuation restoration system based on a sequence-to-sequence model (Seq2Seq) trained on long slices of text which can span multiple sentences.

3 Model Descriptions

We treat correcting run-ons as a sequence labeling task: given a sentence, the model reads each token and learns whether there is a `SPACE` or `PERIOD` following that token, as shown in Table 2. We ap-

This/S shows/S the/S rising/S of/S life/S expectancies/P it/S is/S an/S achievement/S and/S it/S is/S also/S a/S challenge/S ./S

Table 2: NUCLE sentence labeled to indicate what follows each token: a space (`S`) or period (`P`).

ply two sequence models to this task, conditional random fields (*roCRF*) and Seq2Seq (*roS2S*).

3.1 Conditional Random Fields

Our CRF model, roCRF, represents a sentence as a sequence of spaces between tokens, labeled to indicate whether a period should be inserted in that space. Each space is represented by contextual features (sequences of tokens, part-of-speech tags, and capitalization flags around each space), parse features (the highest uncommon ancestor of the word before and after the space, and binary indicators of whether the highest uncommon ancestors are preterminals), and a flag indicating whether the mean per-word perplexity of the text decreases when a period is inserted at the space according to a 5-gram language model.

3.2 Sequence to Sequence Model with Attention Mechanism

Another approach is to treat it as a form of neural sequence generation. In this case, the input sentence is a single run-on sentence. During decoding we pass the binary label which determines if there is terminal punctuation following the token at the current position. We then combine the generated label and the input sequence to get the final output.

Our model, roS2S, is a Seq2Seq attention model based on the neural machine translation model (Bahdanau et al., 2015). The encoder is a bidirectional LSTM, where a recurrent layer processes the input sequence in both forward and backward direction. The decoder is a unidirectional LSTM. An attention mechanism is used to obtain the context vector.

4 Data

Train: Although run-on sentences are common mistakes, existing GEC corpora do not include enough labeled run-on sentences to use as training data. Therefore we artificially generate training examples from a corpus of clean newswire text, Annotated Gigaword (Napoles et al., 2012). We randomly select paragraphs and identify candidate

pairs of adjacent sentences, where the sentences have between 5–50 tokens and no URLs or special punctuation (colons, semicolons, dashes, or ellipses). Run-on sentences are generated by removing the terminal punctuation between the sentence pairs and lowercasing the first word of the second sentence (when not a proper noun). In total we create 2.8 million run-on sentences, and randomly select 1.75M other Gigaword sentences for negative examples. We want the model to learn more patterns of run-on errors by feeding a large portion of positive examples while we report our results on a test set where the ratio is closer to that of real world. We call this data *FakeGiga-Train*. An additional 28k run-ons and 218k non-run-ons are used for validation.

Test: We evaluate on two dimensions: clean versus noisy text and real versus artificial run-ons. In the first evaluation, we artificially generate sentences from Gigaword and NUCLE following the procedure above such that 10% of sentences are run-ons, based on our estimates of their rate in real-world data (similar observations can be found in Watcharapunyawong and Usaha (2012)). We refer to these test sets as *FakeGiga* and *FakeESL* respectively. Please note that the actual run-on sentences in NUCLE are not included in *FakeESL*.

The second evaluation compares performance on artificial versus naturally occurring run-on sentences, using the NUCLE and CoNLL 2013 and 2014 corpora. Errors in these corpora are annotated with corrected text and error types, one of which is *Srun*: run-on sentences and comma splices. *Sruns* occur 873 times in the NUCLE corpus. We found that some of the *Srun* annotations do not actually correct run-on sentences, so we reviewed the *Srun* annotations to exclude any corrections that do not address run-on sentences. We also found that there are 300 out of the 873 sentences with *Srun* annotations which actually perform correction by adding a period. Other *Srun* annotations correct run-ons by converting independent clauses to dependent clauses, but we only target missing periods in this initial work. We manually edit *Srun* annotations so that the only correction performed is inserting periods. (This could be as simple as deleting the comma in the original text of a comma splice or more involved, as in rewriting a dependent clause to an independent clause in the corrected text.) In total, we find fewer than 500 run-on sentences. Run-

	Dataset	RO	Non-RO	Total
Train	FakeGiga-Train	2.76M (61%)	1.75M (39%)	4.51M
Test	FakeGiga	28,232 (11%)	218,076 (89%)	246,308
	RealESL	542 (1%)	58,987 (99%)	59,529
	FakeESL	5,600 (9%)	56,350 (91%)	61,950
	FakeESL-1%	560 (1%)	56,350 (99%)	56,910

Table 3: Number of run-on (RO) and non-run-on (Non-RO) sentences in our datasets.

ning the same procedure over the CoNLL 2013 and 2014 Shared Task test sets results in 59 more run-on sentences and 2,637 non-run-on sentences. We discard all other error annotations and combine the NUCLE train and CoNLL test sets, which we call *RealESL*.

Only 1% of sentences in RealESL are run-ons, which may not be the case in other forms of ESL corpora. So for a fair comparison we down-sample the run-on sentences in FakeESL to form a test set with the same distribution as RealESL, FakeESL-1%. Table 3 describes the size of our data sets.

5 Experiments

Metrics: We report precision, recall, and the $F_{0.5}$ score. In GEC, precision is more important than recall, and therefore the standard metric for evaluation is $F_{0.5}$, which weights precision twice as much as recall.

Baselines: We report results on a balanced random baseline and state-of-the-art models from whole-sentence GEC (NUS18) and punctuation restoration (the Punctuator). NUS18 is the released GEC model of Chollampatt and Ng (2018), trained on two GEC corpora, NUCLE and Lang-8 (Mizumoto et al., 2011). We test two versions of the Punctuator: *Punctuator-EU* is the released model, trained on English Europarl v7 (Koehn, 2005), and *Punctuator-RO*, which we trained on artificial clean data (FakeGiga-Train) using the authors' code.[1]

roCRF: We train our model with $\ell 1$-regularization and $c = 10$ using the CRF++ toolkit.[2] Only features that occur at least 5 times in the training set were included. Spaces are labeled to contain missing punctuation when the marginal probability is less than 0.70. Parameters are tuned to $F_{0.5}$ on 25k held-out sentences.

---

[1]github.com/ottokart/punctuator2
[2]Version 0.59, github.com/taku910/crfpp/

	Clean v. Noisy - Artificial Data						Real v. Artificial - Noisy Data					
	FakeGiga			FakeESL			RealESL			FakeESL-1%		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
Random	0.10	0.10	0.10	0.09	0.09	0.09	0.01	0.01	0.01	0.01	0.01	0.01
Punctuator-EU	0.22	0.45	0.25	0.74	0.48	**0.67**	0.11	**0.65**	0.13	0.12	**0.67**	0.14
Punctuator-RO	0.78	0.57	0.73	0.58	**0.51**	0.56	0.11	0.31	0.13	0.18	0.52	0.21
roCRF	**0.89**	0.49	0.76	**0.83**	0.24	0.55	**0.34**	0.27	**0.32**	**0.32**	0.24	0.30
roS2S	0.84	**0.94**	**0.86**	0.77	0.44	**0.67**	0.30	0.32	0.31	0.30	0.34	**0.31**

Table 4: Performance on clean v. noisy artificial data with 10% run-ons, and real v. artificial data with 1% run-ons.

roS2S: Both the encoder and decoder have a single layer, 1028-dimensional hidden states, and a vocabulary of 100k words. We limit the input sequences to 100 words and use 300-dimensional pre-trained GloVe word embeddings (Pennington et al., 2014). The dropout rate is 0.5 and mini-batches are size 128. We train using Ada-grad with a learning rate of 0.0001 and a decay of 0.5.

6 Results and Analysis

Results are shown in Table 4. A correct judgment is where a run-on sentence is detected and a `PERIOD` is inserted in the right place. Across all datasets, roCRF has the highest precision. We speculate that roCRF consistently has the highest precision because it is the only model to use POS and syntactic features, which may restrict the occurrence of false positives by identifying longer distance, structural dependencies. roS2S is able to generalize better than roCRF, resulting in higher recall with only a moderate impact on precision. On all datasets except RealESL, roS2S consistently has the highest overall $F_{0.5}$ score. In general, Punctuator has the highest recall, probably because it is trained for a more general purpose task and tries to predict punctuation at each possible position, resulting in lower precision than the other models.

NUS18 predicts only a few false positives and no true positives, so $P = R = 0$ and we exclude it from the results table. Even though NUS18 is trained on NUCLE, which RealESL encompasses, its very poor performance is not too surprising given the infrequency of run-ons in NUCLE.

Clean v. Noisy In the first set of experiments (columns 2 and 3), we compare models on clean text (FakeGiga), which has no other grammatical mistakes, and noisy text (FakeESL), which may have several other errors in each sentence. Punctuator-EU is the only model which improves

when tested on the noisy artificial data compared to the clean. It is possible that the speech transcripts used for training Punctuator-EU more closely resemble FakeESL, which is less formal than FakeGiga. All other models do worse, which could be due to overfitting FakeGiga. However, further work is needed to determine how much of the performance drop is due to a domain mismatch versus the frequency of grammatical mistakes in the data.

Real v. Artificial So far, we have only used artificially generated data for training and testing. The second set of experiments (columns 4 and 5) determines if it is easier to correct run-on sentences that are artificially generated compared to those that occur naturally. The Punctuators do poorly on this data because they are too liberal, evidenced by the high recall and very low precision. Our models, roCRF and roS2S, outperform the Punctuators and have similar performance on both the real and artificial run-ons (RealESL and FakeESL-1%). roCRF has significantly higher precision on RealESL while roS2S has significantly higher recall and $F_0.5$ on RealESL and FakeESL-1% (with bootstrap resampling, $p < 0.05$). This supports the use of artificially generated run-on sentences as training data for this task.

7 Conclusions

Correcting run-on sentences is a challenging task that has not been individually targeted in earlier GEC models. We have developed two new models for run-on sentence correction: a syntax-aware CRF model, roCRF, and a Seq2Seq model, roS2S. Both of these outperform leading models for punctuation restoration and grammatical error correction on this task. In particular, roS2S has very strong performance, with $F_{0.5} = 0.86$ and $F_{0.5} = 0.67$ on run-ons generated from clean and noisy data, respectively. roCRF has very high precision $(0.83 \leq P \leq 0.89)$ but low recall, meaning that it

does not generalize as well as the leading system, roS2S.

Run-on sentences have low frequency in annotated GEC data, so we experimented with artificially generated training data. We chose clean newswire text as the source for training data to ensure there were no unlabeled naturally occurring run-ons in the training data. Using ungrammatical text as a source of artificial data is an area of future work. The results of this study are inconclusive in terms of how much harder the task is on clean versus noisy text. However, our findings suggest that artificial run-ons are similar to naturally occurring run-ons in ungrammatical text because models trained on artificial data do just as well predicting real run-ons as artificial ones.

In this work, we found that a leading GEC model (Chollampatt and Ng, 2018) does not correct any run-on sentences, even though there was an overlap between the test and training data for that model. This supports the recent work of Choshen and Abend (2018), who found that GEC systems tend to ignore less frequent errors due to reference bias. Based on our work with run-on sentences, a common error type that is infrequent in annotated data, we strongly encourage future GEC work to address low-coverage errors.

Acknowledgments

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2015. Neural machine translation by jointly learning to align and translate. In Proceedings of the Sixth International Conference on Learning Representations (ICLR).

Xiaoyin Che, Cheng Wang, Haojin Yang, and Christoph Meinel. 2016. Punctuation prediction for unsegmented transcript based on word vector. In Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016), Paris, France. European Language Resources Association (ELRA).

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In AAAI Conference on Artificial Intelligence.

Leshem Choshen and Omri Abend. 2018. Inherent biases in reference-based evaluation for grammatical error correction. In Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 632–642. Association for Computational Linguistics.

Daniel Dahlmeier and Hwee Tou Ng. 2011. Grammatical error correction with alternating structure optimization. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 915–923, Portland, Oregon, USA. 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, Atlanta, Georgia. Association for Computational Linguistics.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. In Proceedings of the Eighteenth Conference on Computational Natural Language Learning: Shared Task, pages 15–24, Baltimore, Maryland. Association for Computational Linguistics.

Agustin Gravano, Martin Jansche, and Michiel Bacchiani. 2009. Restoring punctuation and capitalization in transcribed speech. In Proceedings of the International Conference on Acoustics, Speech and Signal Processing (ICASSP), pages 4741–4744. IEEE.

Ross Israel, Joel Tetreault, and Martin Chodorow. 2012. Correcting comma errors in learner essays, and restoring commas in newswire text. In Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 284–294, Montréal, Canada. 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, Austin, Texas. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers), pages 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Dona M Kagan. 1980. Run-on and fragment sentences: An error analysis. Research in the Teaching of English, 14(2):127–138.

Philipp Koehn. 2005. Europarl: A parallel corpus for statistical machine translation. In MT summit, volume 5, pages 79–86.

Claudia Leacock, Martin Chodorow, Michael Gamon, and Joel Tetreault. 2014. Automated grammatical error detection for language learners. Synthesis lectures on human language technologies, 7(1):1–170.

John Lee, Chak Yan Yeung, and Martin Chodorow. 2014. Automatic detection of comma splices. In Proceedings of the 28th Pacific Asia Conference on Language, Information, and Computation, pages 551–560, Phuket,Thailand. Department of Linguistics, Chulalongkorn University.

Wei Lu and Hwee Tou Ng. 2010. Better punctuation prediction with dynamic conditional random fields. In Proceedings of the 2010 Conference on Empirical Methods in Natural Language Processing, pages 177–186, Cambridge, MA. Association for Computational Linguistics.

Nitin Madnani, Joel Tetreault, and Martin Chodorow. 2012. Exploring grammatical error correction with not-so-crummy machine translation. In Proceedings of the Seventh Workshop on Building Educational Applications Using NLP, pages 44–53. Association for Computational Linguistics.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In Proceedings of 5th International Joint Conference on Natural Language Processing, pages 147–155, Chiang Mai, Thailand. Asian Federation of Natural Language Processing.

Courtney Napoles, Matthew Gormley, and Benjamin Van Durme. 2012. Annotated Gigaword. In Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX), pages 95–100, Montréal, Canada. 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.

Stephan Peitz, Markus Freitag, Arne Mauser, and Hermann Ney. 2011. Modeling punctuation prediction as machine translation. In International Workshop on Spoken Language Translation (IWSLT) 2011.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. Glove: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2011. Algorithm selection and model adaptation for ESL correction tasks. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 924–933, Portland, Oregon, USA. Association for Computational Linguistics.

Joel Tetreault, Jennifer Foster, and Martin Chodorow. 2010. Using parse features for preposition selection and error detection. In Proceedings of the ACL 2010 Conference Short Papers, pages 353–358, Uppsala, Sweden. Association for Computational Linguistics.

Ottokar Tilk and Tanel Alumäe. 2016. Bidirectional recurrent neural network with attention mechanism for punctuation restoration. In Proceedings of Interspeech, pages 3047–3051.

Somchai Watcharapunyawong and Siriluck Usaha. 2012. Thai efl students writing errors in different text types: The interference of the first language. English Language Teaching, 6(1):67.

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.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

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.