Erroneous data generation for Grammatical Error Correction

Shuyao Xu **Jiehao Zhang** **Jin Chen** **Long Qin**
Singsound Inc.
{xushuy,zhangjiehao,chenjin,qinlong}@singsound.com

Abstract

It has been demonstrated that the utilization of a monolingual corpus in neural Grammatical Error Correction (GEC) systems can significantly improve the system performance. The previous state-of-the-art neural GEC system is an ensemble of four *Transformer* models pretrained on a large amount of Wikipedia Edits. The Singsound GEC system follows a similar approach but is equipped with a sophisticated erroneous data generating component. Our system achieved an $F_{0.5}$ of 66.61 in the BEA 2019 Shared Task: Grammatical Error Correction. With our novel erroneous data generating component, the Singsound neural GEC system yielded an M^2 of 63.2 on the CoNLL-2014 benchmark (8.4% relative improvement over the previous state-of-the-art system).

1 Introduction

The most effective approaches to Grammatical Error Correction (GEC) task are machine translation based methods. Both Statistical Machine Translation (SMT) approaches and Neural Machine Translation (NMT) methods have achieved promising results in the GEC task.

Pretraining a decoder as a language model is an effective method to improve the performance of neural GEC systems (Junczys-Dowmunt et al., 2018). As an extension of this work, Lichtarge et al. (2018) showed pretraining on 4 billion tokens of Wikipedia edits to be beneficial for the GEC task.

In this work, we investigate a similar approach by systematically generating parallel data for pretraining. As shown in Table 1, in addition to spelling errors (price → puice), transposition errors (independent voters → voters independent) and concatenation errors (the man → theman), our method also introduces errors such as ramped → ramping. Our approach obtained competitive results compared to the top systems in the BEA 2019 GEC Shared Task. Both our single model and ensemble models have exceeded the previous state-of-the-art systems on the CoNLL-2014 (Ng et al., 2014) benchmark and our system reaches human-level performance on the JFLEG (Napoles et al., 2017) benchmark.

Origin	the primary is **open** to **independent voters** .
Generated	the primary is **opens** to **voters independhent** .
Origin	the **price** of alcohol is **ramped** up at every budget .
Generated	the **puice** of alchool is **ramping** up at every budget .
Origin	they say the **police** shot and killed **the man** after he had **fired** at them .
Generated	they say the **polices** shot and killed **theman** after he had **firing** at them .

Table 1: Examples of generated data.

2 Related Work

Chollampatt and Ng (2018) used a convolutional sequence-to-sequence (seq2seq) model (Gehring et al., 2017) with a large language model for rescoring. Their model was the first NMT based GEC system that exceeded the strong SMT baseline system (Junczys-Dowmunt and Grundkiewicz, 2016) which combined a Phrase-based Machine Translation (PBMT) with a large language model. Then a hybrid PBMT-NMT system (Grundkiewicz and Junczys-Dowmunt, 2018) appeared to reach the new state-of-the-art on the CoNLL-2014 benchmark. Later, various pure neu-

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 149–158
Florence, Italy, August 2, 2019.

Corpus	Sentences	Tokens	Anno.
WMT11	115M	2362M	No
1B words	30M	769M	No
Lang-8	1037K	12M	Yes
NUCLE	57K	1.2M	Yes
FCE	28K	455K	Yes
ABCN	34K	628K	Yes

Table 2: Statistics for training data sets.

Corpus	Sentences	Scorer
ABCN dev	4384	ERRANT
ABCN test	4477	ERRANT
JFLEG test	747	GLEU
CoNLL-2014 test	1312	M^2 Scorer

Table 3: Statistics for test and development data.

Length	Err.	Prob.	Length	Err.	Prob.
$[1, 3)$	0	0.50	$[6, 9)$	2	0.30
	1	0.50		3	0.45
$[3, 6)$	1	0.50		4	0.25
	2	0.50	$[16, 20)$	3	0.10
$[9, 16)$	3	0.15		4	0.15
	4	0.25		5	0.15
	5	0.30		6	0.30
	6	0.30		7	0.30
$[20, 30)$	4	0.10	$[30, \infty)$	5	0.10
	5	0.15		6	0.15
	6	0.15		7	0.15
	7	0.30		8	0.30
	8	0.30		9	0.30

Table 4: Probability distribution of sentence errors.

ral systems (Ge et al., 2018; Junczys-Dowmunt et al., 2018; Lichtarge et al., 2018) reported state-of-the-art results successively. Ge et al. presented the fluency boosting method which was demonstrated to be effective to improve performance of GEC seq2seq models. The system proposed by Junczys-Dowmunt et al. (2018) is an ensemble of *Transformer* models (Vaswani et al., 2017); they pretrained the decoder of transformer as a language model on a large monolingual corpus. To our best knowledge, the current state-of-the-art GEC system on both the CoNLL-2014 benchmark and the JFLEG benchmark is the system presented by Lichtarge et al. (2018), which is an ensemble of four *Transformer* models pretrained on Wikipedia revisions and then fine-tuned on Lang-8 (Mizumoto et al., 2011).

3 Data

We list the training data in Table 2. The text data used to generate parallel corpus automatically was the One Billion Words Benchmark dataset (1B words) (Chelba et al., 2013) and the WMT11 monolingual corpus (WMT11) which can be obtained from WMT11 Website[1]. Our fine-tuning data is Lang-8 (Mizumoto et al., 2011; Tajiri et al., 2012), NUS Corpus of Learner English (NUCLE) (Dahlmeier et al., 2013), FCE (Yannakoudakis et al., 2011), the Cambridge English Write & Improve (W&I) corpus and the LOCNESS corpus (ABCN) (Granger, 1998; Bryant et al., 2019) .

[1] `http://statmt.org/wmt11/training-monolingual.tgz`

Table 3 shows the development and test data sets in our experiments. We choose the ABCN dev set as our development set and the ABCN test, the CoNLL-2014 test, the JFLEG as our benchmark. For these benchmarks, we report precision (P), recall (R) and $F_{0.5}$ with ERRANT (Bryant et al., 2017) on the ABCN test, GLEU (Sakaguchi et al., 2016) on the JFLEG test set (Napoles et al., 2017). To compare with previous state-of-the-art systems, we provide results of MaxMatch (M^2) Scorer (Dahlmeier and Ng, 2012) on the CoNLL-2014 test set.

4 Erroneous Data Generation

In this section, we describe our error generating method. For each sentence, we assign a probability distribution (as shown in Table 4) to determine the number of errors according to the sentence length. The parameters in Table 4 are determined empirically, as well as the parameters in Table 5, Table 6 and Table 7. Because of the time limitation of the GEC competition, we did not optimize these parameters.

After the number of errors (E) in a sentence has been determined, we randomly select E tokens from all the tokens of the sentence with equal probability to be errors. And for each error, we apply a random variable (Table 5) to determine which error type it should be.

We introduce five error types:

- Concatenation: combine two consecutive tokens, *e.g.*, hello world → helloworld.
- Misspelling: introduce spelling errors into

Type	Prob.
Concatenation	0.12
Misspell	0.45
Substitution	0.40
Deletion	0.00
Transposition	0.03

Table 5: Error types.

Tok. length	Err.	Prob.
$[1, 3)$	0	1.00
$[3, 5)$	1	1.00
$[5, 10)$	1	0.80
	2	0.20
$[10, \infty)$	1	0.75
	2	0.15
	3	0.10

Table 6: Number of misspells in a token.

Type	Prob.
Deletion	0.30
Insertion	0.15
Transposition	0.25
Replacement	0.30

Table 7: Mispell types.

words, *e.g.*, computer → camputer.

- Substitution: we introduce seven different types of substitutions.
- Deletion: delete the token.
- Transposition: the token exchange position with a consecutive token.

4.1 Misspelling

To generate misspellings, we introduce a random variable to determine how many errors in the token according to the token length (parameters are shown in Table 6.), and we randomly insert errors into the token.

For each spelling error, we apply another random variable to determine which error type should be. We introduce four spelling error types (Table 7 lists the parameters.).

- Deletion: delete the character.
- Insertion: insert a random English letter into the current position.
- Transposition: exchange position with the consecutive character.
- Replacement: replace the current character with a random English character.

We only introduce spelling errors into words belonging to a vocabulary list of 32k ordinary words[2] which does not include numerals (*e.g.*, 2019), tokens that contain digits (*e.g.*, Lang8), URLs or non-word symbols (*e.g.*, $\geq \bigtriangledown \leq$).

4.2 Substitution

We introduce seven types of substitutions according to token and its part-of-speech (POS).

- Substitution between Prepositions. *E.g.*, in, on, at, through, for, with.
- Substitution between Articles. *E.g.*, a, an, the.
- Substitution between Pronouns (Singular). *E.g.*, he, she, his, him, her, hers.
- Substitution between Pronouns (Plural). *E.g.*, their, them, they, theirs.
- Substitution between Wh words. *E.g.*, which, where, what, how, when, who, whose, whom.
- Substitution between Modal verbs. *E.g.*, will, shall, can, may, would, could, might.
- Substitution in a Word Tree (see 4.3 for details).

4.3 Word Tree

We want to make substitutions such as going → gone, useful → usable, administration → administrative. To make such substitution possible, we introduce the Word Tree.

A Word Tree represents a group of words that share the same stem but have different suffixes. Figure 1 shows an example of Word Tree of "use". A node denotes a word (*e.g.*, usable) and corresponding Extended part-of-speech (EPOS) (*e.g.*, VBP_JJ_BLE) (see 4.4 for details.), and an edge indicates the root from which the word is derived (*e.g.*, "usable" is derived from "use").

With EPOS, we can easily set rules or assign probability distributions to determine which substitutions are more likely to happen, (*e.g.*, singular ↔ plural, VBD ↔ VBZ ↔ VBP ↔ VBN ↔

[2]We manually created this vocabulary for building the Word Tree (see 4.3 for details).

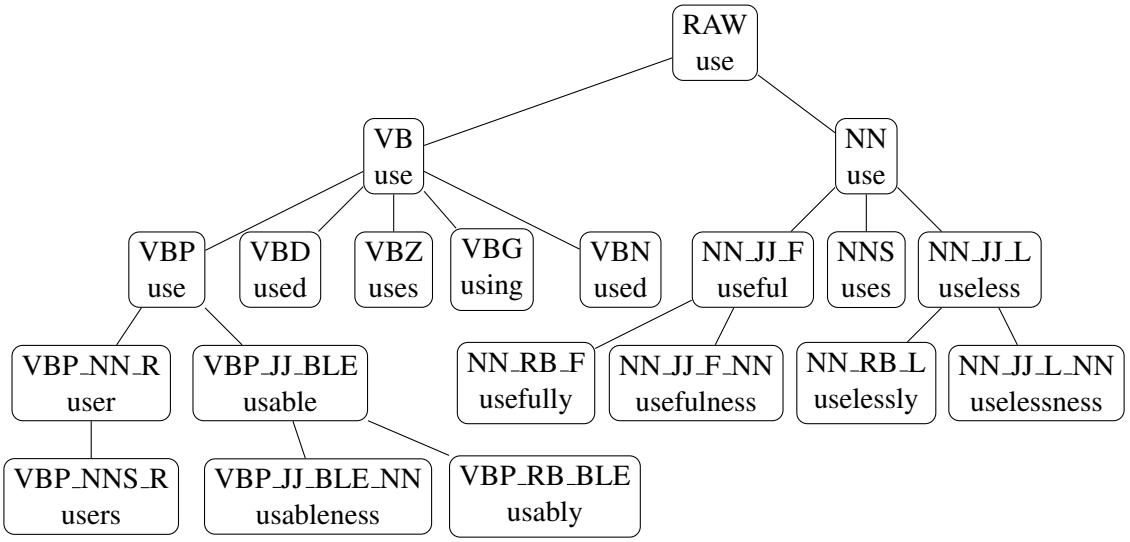

Figure 1: Word Tree: use

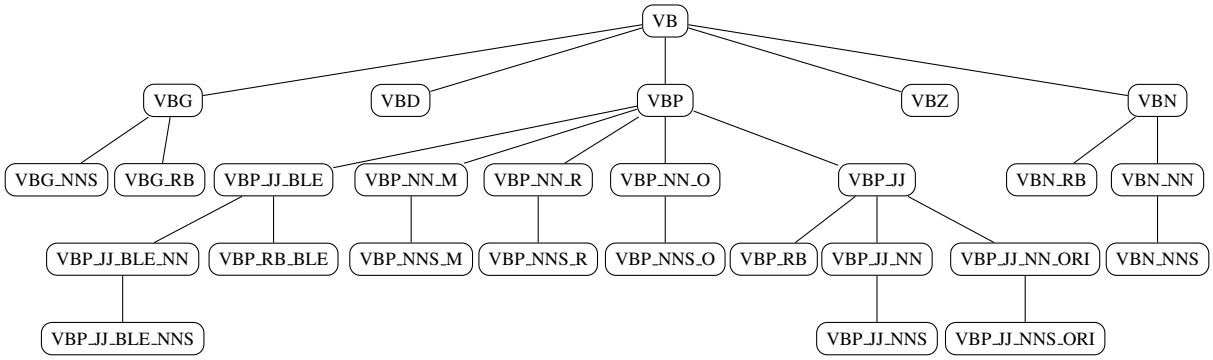

Figure 2: Verb branch of the EPOS Tree

VBG, adjective $\leftrightarrow$ adverb), and which substitutions are less likely to happen (*e.g.*, happiest JJS $\leftrightarrow$ happiness JJ_NN). In our experiments, due to the time limitation of the competition, we simply assigned a uniform distribution to all existing words in a Word Tree, excluding substitutions that were definitely unlikely to occur such as substitutions between the words in an NN_JJ_F branch (*e.g.*, careful) and the words in an NN_JJ_L branch (*e.g.*, carelessness).

4.4 Extended part-of-speech

A Word Tree can contain multiple words of the same POS. As shown in the example in Figure 1, use, user and usefulness can all be nouns. Therefore, in order to identify the different roles for words in a Word Tree, we propose EPOS, derived from part-of-speech (POS) and the surface form of the word.

POS explains how a word is used (mostly syntactically) in a sentence. Compared to POS, EPOS also reflects some semantic role of a word in a sentence.

We define EPOS in Table 11 in the Appendix. We used NLTK (Bird, 2006) as our POS tagger, and use NLTK-style tags in this paper.

We briefly describe our method of creating word trees.

a. Extract the vocabulary from a text corpus which is tagged with NLTK POS tagger.

b. Create three tables for Noun, Verb, and Adjective respectively. The Noun table contains two columns: singular and plural; the Verb table has six columns: original verb form (VB), non-third person present (VBP), third person present (VBZ), past tense (VBD), past participle (VBN) and present participle (VBG); the Adjective table has four columns: adjective, adverb, comparative degree and superlative de-

#	System	P	R	$F_{0.5}$
1	UEDIN-MS	72.28	60.12	69.47
	...			
4	CAMB-CLED	70.49	55.07	66.75
5	**Singsound**	70.17	55.39	66.61

Table 8: Results of BEA 2019 GEC competition.

	dev	test		
	$F_{0.5}$	P	R	$F_{0.5}$
single	52.29	66.06	56.68	63.94
w/o pretrain	44.60	50.59	43.60	49.02
4 ensemble	55.37	70.14	57.57	67.21
w/o pretrain	47.01	56.05	44.33	53.24

Table 9: Results of ABCN set. "w/o pretrain" refers to models **without** pretraining.

gree. Then we fill words into corresponding entries according to their POS tags. Words that cannot be filled in any of the above tables are filled into a list.

c. Manually check and correct all entries of the three tables, and fill missing entries as well.

d. Define EPOS as listed in Table 11 in the Appendix according to suffix transforming rules.

e. Extract a RAW list from the vocabulary according to the suffix transforming rules.

f. Create an EPOS tree structure for each token in the RAW list, and then fill each word from the vocabulary into the corresponding entry of the corresponding EPOS tree (The full structure of the EPOS Tree is described in Table 12 in the Appendix, and Figure 2 shows the Verb branch); then prune empty entries in the trees.

g. Manually check every entry of every word tree, and fix all incorrect entries.

h. Update the defined EPOS (final version in Table 11) and the EPOS tree (Table 12); recreate word trees.

i. Repeat step g and h until satisfied.

5 Experiments

In our experiments, we generated a corpus of 3 billion tokens, of which about 24% were errors.

Following Lichtarge et al. (2018), we also use *Transformer* as our encoder-decoder model, using *Tensor2Tensor* open source implementation [3].

The models are trained on words, and rare words are segmented into sub-words with the byte pair encoding (BPE) (Sennrich et al., 2015). We use 6 layers for both encoder and decoder, and 4 attention heads. The embedding size and hidden size are 1024, and the filter size for all position-wise feed forward network is 4096. We set dropout rate to 0.3, and source word dropout is set to 0.2 as a noising technique. Following Junczys-Dowmunt et al. (2018), source, target and output embeddings are tied in our models.

Following Lichtarge et al. (2018), we first trained our model on an artificially generated parallel corpus with a batch size of approximately 3072 tokens. Then we set the batch size to 2048 tokens and fine-tuned on human annotated data for 20 epochs, and we averaged the 5 best checkpoints. Finally, the averaged model was fine-tuned on the ABCN and FCE training data for 1000 steps as domain adaptation (Junczys-Dowmunt et al., 2018).

There are about 50% sentence pairs without any correction in the Lang-8 dataset, and we noticed that training with too many error-free sentence pairs had a negative effect. Therefore, we filtered out these error-free sentence pairs in the Lang-8 dataset. Since the NUCLE, FCE and ABCN datasets are much smaller than the Lang-8 set, we did not filter out the error-free sentence pairs in these datasets.

We used beam search for decoding with a beam size of 4 at evaluation time. For the ensemble, we averaged logits from 4 *Transformer* models with identical hyper-parameters at each decoding step. Following (Grundkiewicz and Junczys-Dowmunt, 2018; Junczys-Dowmunt et al., 2018; Lichtarge et al., 2018), we preprocessed the JFLEG dataset with spell-checking. We did not apply spell-checking to the ABCN and CoNLL-2014 datasets.

6 Results and Discussion

The results of the Singsound System in the GEC competition (Table 8) were obtained by an ensemble of four models. Because of the time limitation, we only trained two independent models

[3] https://github.com/tensorflow/tensor2tensor

	Model	CoNLL-2014			CoNLL-10 (SvH)			JFLEG
		P	R	$F_{0.5}$	P	R	$F_{0.5}$	GLEU
(1)	Word&Char SMT-GEC	62.7	33.0	53.1			68.3	56.8
(2)	MLConv (4 ensemble)	65.5	33.1	54.8				57.5
(3)	Transformer (single)			53.0				57.9
	Transformer (4 ensemble)	63.0	38.9	56.1				58.5
	Transformer (4 ensemble) + LM	61.9	40.2	55.8				59.9
(4)	Hybrid PBMT+NMT+LM	66.8	34.5	56.3	83.2	47.0	72.0	61.5
(5)	Transformer (single)	62.2	37.8	54.9				59.3
	Transformer (4 ensemble)	67.5	37.8	58.3				62.4
Singsound	Transformer (single)	68.3	**42.5**	60.9	83.5	55.2	75.7	60.8
	Transformer (4 ensemble)	**73.0**	41.1	**63.2**	**86.0**	53.8	**76.8**	**62.6**
	Human avg.				73.5	**69.6**	72.6	62.4

Table 10: Comparison with top performing systems on CoNLL and JFLEG datasets. (1): Chollampatt and Ng (2017) (2): Chollampatt and Ng (2018); (3): Junczys-Dowmunt et al. (2018); (4): Grundkiewicz and Junczys-Dowmunt (2018); (5): Lichtarge et al.(2018).

from scratch. The other two were based on existing trained models. Concretely, after we got a model trained from scratch, we kept training it on the generated corpus for 0.2 epoch; then fine-tuned the model on the annotated data and ABCN and FCE training sets as before.

We provide the performance of our single model and the ensemble of 4 independently trained models [4] on the ABCN dev and test datasets in Table 9. As the results shown in Table 9, models pretrained on the generated corpus significantly outperform the models without pretraining.

To compare with previous state-of-the-art GEC systems, we evaluated our systems on the CoNLL-2014 and JFLEG datasets. As the results shown in Table 10, our single model exceeded previous state-of-the-art systems on the CoNLL-2014 dataset. Our ensemble models achieved 8.4% relative improvement over the latest state-of-the-art results on the CoNLL-2014 benchmark.

We also report the results on the CoNLL-2014 10 annotation dataset (denoted as CoNLL-10) (Bryant and Ng, 2015) which is an extension of the CoNLL-2014 test set with 10 annotators. The human-level scores are calculated by averaging the scores for each annotator with regard to the remaining annotators. Following Chollampatt and Ng (2017), scores on CoNLL-10 (SvH) are calculated by removing one set of human annotations at a time and evaluating the system against the remaining sets. Our models reach human-level performance on both CoNLL-10 and JFLEG benchmarks.

[4]The four models are trained on the same data with the same hyper-parameter set.

7 Conclusion

In this work, we present a novel erroneous data generating method for training English GEC models. Our experiments show that *Transformer* models pretrained on generated corpus significantly outperform the previous GEC systems that are also based on *Transformer*. We also present a novel tool: the Word Tree, which represents a group of words that share the same stem but have different suffixes; and we show that one possible application of the Word Tree is generating erroneous text for training GEC models.

Acknowledgments

We thank Xiaoxue Fan, Miao Xue and Yueming Gao for their help in checking the three tables required for creating the Word Trees.

References

Steven Bird. 2006. NLTK: The natural language toolkit. *meeting of the Association for Computational Linguistics*, pages 69–72.

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 Edward John Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. Association for Computational Linguistics.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction. 1:697–707.

Ciprian Chelba, Tomas Mikolov, Mike Schuster, Qi Ge, Thorsten Brants, Phillipp Koehn, and Tony Robinson. 2013. One billion word benchmark for measuring progress in statistical language modeling. *Computer Science*.

Shamil Chollampatt and Hwee Tou Ng. 2017. Connecting the dots: Towards human-level grammatical error correction. pages 327–333.

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

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.

Tao Ge, Furu Wei, and Ming Zhou. 2018. 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.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. 2017. Convolutional sequence to sequence learning. In *Proceedings of the 34th International Conference on Machine Learning-Volume 70*, pages 1243–1252. JMLR. org.

Sylviane Granger. 1998. *The computer learner corpus: a versatile new source of data for SLA research*. na.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *arXiv preprint arXiv:1804.05945*.

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

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *arXiv preprint arXiv:1804.05940*.

Jared Lichtarge, Christopher Alberti, Shankar Kumar, Noam Shazeer, and Niki Parmar. 2018. Weakly supervised grammatical error correction using iterative decoding. *arXiv preprint arXiv:1811.01710*.

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.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

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.

Keisuke Sakaguchi, Courtney Napoles, Matt Post, and Joel R Tetreault. 2016. Reassessing the goals of grammatical error correction: Fluency instead of grammaticality. *Transactions of the Association for Computational Linguistics*, 4(1):169–182.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2015. Neural machine translation of rare words with subword units. *arXiv preprint arXiv:1508.07909*.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for esl learners using global context. pages 198–202.

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.

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-Volume 1*, pages 180–189. Association for Computational Linguistics.

A Appendix

EPOS	POS	Annotation	Examples
RAW		Root of word trees, original form	use
NN	NN	Noun	use
NNS	NNS	Plural form of Noun	uses
NN_JJ_F	JJ	NN + ful [5]	useful
NN_JJ_F_NN	NN	NN_JJ_F + ness	usefulness
NN_JJ_F_NNS	NNS	Plural form of NN_JJ_F_NN	
NN_JJ_F_NN_ORI	NN	Adjective used as Noun	dreadful
NN_JJ_F_NNS_ORI	NNS	Plural form of NN_JJ_F_NN_ORI	dreadfuls
NN_RB_F	RB	Adverb form of NN_JJ_F	usefully
NN_JJ_L	JJ	NN + less	useless
NN_JJ_L_NN	NN	NN_JJ_L + ness	uselessness
NN_JJ_L_NNS	NNS	Plural form of NN_JJ_L_NN	
NN_JJ_L_NN_ORI	NN	Adjective used as Noun	wireless
NN_JJ_L_NNS_ORI	NNS	Plural form of NN_JJ_L_NN_ORI	wirelesses
NN_RB_L	RB	Adverb form of NN_JJ_L	uselessly
NN_JJ_OUS	JJ	NN + ous	dangerous
NN_JJ_OUS_NN	NN	NN_JJ_OUS + ness	dangerousness
NN_JJ_OUS_NNS	NNS	Plural form of NN_JJ_OUS_NN	
NN_RB_OUS	RB	Adverb form of NN_JJ_OUS	dangerously
NN_JJ_AL	JJ	NN + al	rational
NN_JJ_AL_NN	NN	NN_JJ_AL + ness	rationalness
NN_JJ_AL_NNS	NNS	Plural form of NN_JJ_AL_NN	
NN_RB_AL	RB	Adverb form of NN_JJ_AL	rationally
NN_JJ_Y	JJ	NN + y	lucky
NN_JJR_Y	JJR	Comparative degree of NN_JJ_Y	luckier
NN_JJS_Y	JJS	Superlative degree of NN_JJ_Y	luckiest
NN_JJ_Y_NN	NN	NN_JJ_Y + ness	luckiness
NN_JJ_Y_NNS	NNS	Plural form of NN_JJ_Y_NN	
NN_JJ_Y_NN_ORI	NN	Adjective used as Noun	safety
NN_JJ_Y_NNS_ORI	NNS	Plural form of NN_JJ_Y_NN_ORI	safeties
NN_RB_Y	RB	Adverb form of NN_JJ_Y	luckily
NN_JJ_D	JJ	NN + ed	warmhearted
NN_JJ_D_NN	NN	NN_JJ_D + ness	warmheartedness
NN_JJ_D_NNS	NNS	Plural form of NN_JJ_D_NN	
NN_RB_D	RB	Adverb form of NN_JJ_D	warmheartedly
VB	VB	Original form of verbs	go
VBD	VBD	Past tense	went
VBZ	VBZ	Present third person singular	goes
VBN	VBN	Past participle	gone
VBN_NN	NN	VBN + ness	limitedness
VBN_NNS	NNS	Plural form of VBN_NN	
VBN_NNS_ORI	NNS	Plural form of VBN when VBN used as Noun	shots, thoughts
VBN_RB	RB	Adverb form of VBN	excitedly
VBG	VBG	Present participle	baking
VBG_NNS_ORI	NNS	Plural form of VBG when VBG used as Noun	bakings
VBG_RB	RB	Adverb form of VBG	excitingly
VBP	VBP	non-third person present	go

VBP_NN_O	NN	VBP + ion	connection
VBP_NNS_O	NNS	Plural form of VBP_NN_O	connections
VBP_NN_R	NN	VBP + er / or / ar	dancer, editor
VBP_NNS_R	NNS	Plural form of VBP_NN_R	dancers, editors
VBP_JJ_BLE	JJ	VBP + able / ible	usable
VBP_JJ_BLE_NN	NN	VBP_JJ_BLE + ness	usableness
VBP_JJ_BLE_NNS	NNS	Plural form of VBP_JJ_BLE_NN	
VBP_RB_BLE	RB	Adverb form of VBP_JJ_BLE	usably
VBP_JJ	JJ	VBP + ive	active
VBP_RB	RB	Adverb form of VBP_JJ	actively
VBP_JJ_NN	NN	VBP_JJ + ness	attractiveness
VBP_JJ_NNS	NNS	Plural form of VBP_JJ_NN	
VBP_JJ_NN_ORI	NN	VBP_JJ used as Noun	representative
VBP_JJ_NNS_ORI	NNS	Plural form of VBP_JJ_NN_ORI	representatives
VBP_NN_M	NN	VBP + ment	movement
VBP_NNS_M	NNS	Plural form of VBP_NN_M	movements
JJ	JJ	Adjectivel	happy
JJS	JJS	Superlative degree of Adjective	happiest
JJR	JJR	Comparative degree of Adjective	happier
JJ_NN	NN	JJ + ness	happiness
JJ_NNS	NNS	Plural form of JJ_NN	happinesses
RB	RB	Adverb	happily
RBR	RBR	Comparative degree of Adverb	harder
RBS	RBS	Superlative degree of Adverb	hardest
CD	CD	Cardinal digits	one
CD_JJ	JJ	Adjective form of CD	first
CD_RB	RB	Adverb form of numbers	firstly
CD_JJ_NN_ORI	NN	Adjective used as Noun	first
CD_JJ_NNS_ORI	NNS	Plural form of CD_JJ_NN_ORI	firsts
CD_RB_ORI	RB	Adverbs that are same as CD_JJ	first
CD_NNS	NNS	Plural form of CD	ones
DT	DT	Determiner	the
WRB	WRB	Wh-adverb	how, where
PRP	PRP	Personal pronoun	I, you, they
IN	IN	Preposition or subordinating conjunction	in, from, after
CC	CC	Coordinating conjunction	and
MD	MD	Modal verb	can
OFS		Any POS out of the POS column	

Table 11: EPOS table.

[5]By abuse notation, ”+” denotes ”with some suffix”.

Parent	Children
RAW	NN, JJ, VB, IN, OFS, CC, MD, DT, PRP, CD, WDT, WP, WRB
NN	NNS, NN_JJ_F, NN_JJ_L, NN_JJ_Y, NN_JJ_D, NN_JJ_OUS, NN_JJ_AL
NN_JJ_D	NN_RB_D, NN_JJ_D_NN
NN_JJ_Y	NN_RB_Y, NN_JJR_Y, NN_JJS_Y, NN_JJ_Y_NN, NN_JJ_Y_NN_ORI
NN_JJ_Y_NN	NN_JJ_Y_NNS
NN_JJ_Y_NN_ORI	NN_JJ_Y_NNS_ORI
NN_JJ_F	NN_RB_F, NN_JJ_F_NN, NN_JJR_F, NN_JJS_F, NN_JJ_F_NN_ORI
NN_JJ_F_NN	NN_JJ_F_NNS
NN_JJ_F_NN_ORI	NN_JJ_F_NNS_ORI
NN_JJ_L	NN_RB_L, NN_JJ_L_NN, NN_JJR_L, NN_JJS_L, NN_JJ_L_NN_ORI
NN_JJ_L_NN	NN_JJ_L_NNS
NN_JJ_L_NN_ORI	NN_JJ_L_NNS_ORI
NN_JJ_AL	NN_RB_AL, NN_JJ_AL_NN
NN_JJ_AL_NN	NN_JJ_AL_NNS
NN_JJ_OUS	NN_RB_OUS, NN_JJ_OUS_NN
NN_JJ_OUS_NN	NN_JJ_OUS_NNS
VB	VBP, VBD, VBZ, VBG, VBN
VBP	VBP_JJ, VBP_NN_R, VBP_NN_M, VBP_NN_O, VBP_JJ_BLE
VBP_JJ	VBP_RB, VBP_JJ_NN, VBP_JJ_NN_ORI
VBP_JJ_NN	VBP_JJ_NNS
VBP_JJ_NN_ORI	VBP_JJ_NNS_ORI
VBP_JJ_BLE	VBP_RB_BLE VBP_JJ_BLE_NN
VBP_JJ_BLE_NN	VBP_JJ_BLE_NNS
VBP_NN_R	VBP_NNS_R
VBP_NN_M	VBP_NNS_M
VBP_NN_O	VBP_NNS_O
VBG	VBG_RB, VBG_NNS
VBN	VBN_RB, VBN_NN
VBN_NN	VBN_NNS
JJ	JJR, JJS, RB, JJ_NN, JJ_NN_ORI
JJ_NN	JJ_NNS
JJ_NN_ORI	JJ_NNS_ORI
RB	RBR, RBS
CD	CD_JJ, CD_JJ_NN_ORI, CD_NNS
CD_JJ	CD_RB, CD_RB_ORI
CD_JJ_NN_ORI	CD_JJ_NNS_ORI

Table 12: Structure of the EPOS Tree.