ALLECS: A Lightweight Language Error Correction System

Muhammad Reza Qorib, Geonsik Moon, and **Hwee Tou Ng**
Department of Computer Science, National University of Singapore
mrqorib@comp.nus.edu.sg, moon97@nus.edu.sg, nght@comp.nus.edu.sg

Abstract

In this paper, we present ALLECS, a lightweight web application to serve grammatical error correction (GEC) systems so that they can be easily used by the general public. We design ALLECS to be accessible to as many users as possible, including users who have a slow Internet connection and who use mobile phones as their main devices to connect to the Internet. ALLECS provides three state-of-the-art base GEC systems using two approaches (sequence-to-sequence generation and sequence tagging), as well as two state-of-the-art GEC system combination methods using two approaches (edit-based and text-based). ALLECS can be accessed at https://sterling8.d2.comp.nus.edu.sg/gec-demo/[1].

1 Introduction

English has become the *de facto* language for international discourse, spoken by approximately more than 1.4 billion speakers, with almost 75% of them being non-native speakers (Eberhard et al., 2022). As the number of English-as-a-second-language (ESL) and English-as-a-foreign-language (EFL) speakers keeps increasing, the need for automated tools to assist ESL and EFL speakers in learning and writing English also increases in tandem.

Grammatical Error Correction (GEC) is a task that aims to automatically detect and correct errors that are present in a text, including grammatical errors, orthographic errors, misspellings, word choice errors, etc. (Ng et al., 2014; Bryant et al., 2022). GEC tools have a wide range of applications, including helping native speakers to correct their occasional mistakes, assisting language learners (Knutsson et al., 2003; Chollampatt et al., 2016; Nadejde and Tetreault, 2019; Katinskaia and Yangarber, 2021), and improving the quality of other natural language processing (NLP) tasks (Yin et al., 2020; Liao et al., 2022).

GEC experienced significant progress in the last decade thanks to the HOO (Dale and Kilgarriff, 2011), CoNLL-2013 (Ng et al., 2013), CoNLL-2014 (Ng et al., 2014), and BEA-2019 (Bryant et al., 2019) shared tasks. Qorib and Ng (2022) reported that state-of-the-art GEC systems have exceeded human-level performance based on the standard evaluation metric, $F_{0.5}$ score. With the rapid progress in GEC, different approaches emerged to achieve state-of-the-art performance, including system combination (Qorib et al., 2022), sequence tagging (Lai et al., 2022), and sequence-to-sequence generation (Rothe et al., 2021). Even though many GEC systems publicly publish their source code along with their trained models, these systems can typically only be run through a command-line interface on a highly capable computing resource. Command-line interface is not easy to navigate for non-technical people and few people have access to a capable computing resource. These factors become a formidable barrier for the general public to benefit from the research progress of GEC.

In this paper, we present ALLECS (A Lightweight Language Error Correction System), a simple system to release GEC models to the general public with a lightweight web-based interface. Our web interface only requires 2.5 KB of data transfer overhead for each run. This means that ALLECS can be readily used by users in developing countries with slow Internet connections (Delaporte and Bahia, 2020). Furthermore, we use a responsive design for the web interface, allowing it to be run comfortably on devices with various screen dimensions and sizes. This allows users to use the system through a mobile phone, which is the dominant device type to connect to the Internet in developing countries (Glushkova et al., 2019). Usability is important for people living in developing countries, as they are the ones who can benefit the most from

[1]The source code and a video demonstration of ALLECS can be accessed at https://github.com/nusnlp/ALLECS.

Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics
System Demonstrations, pages 298–306

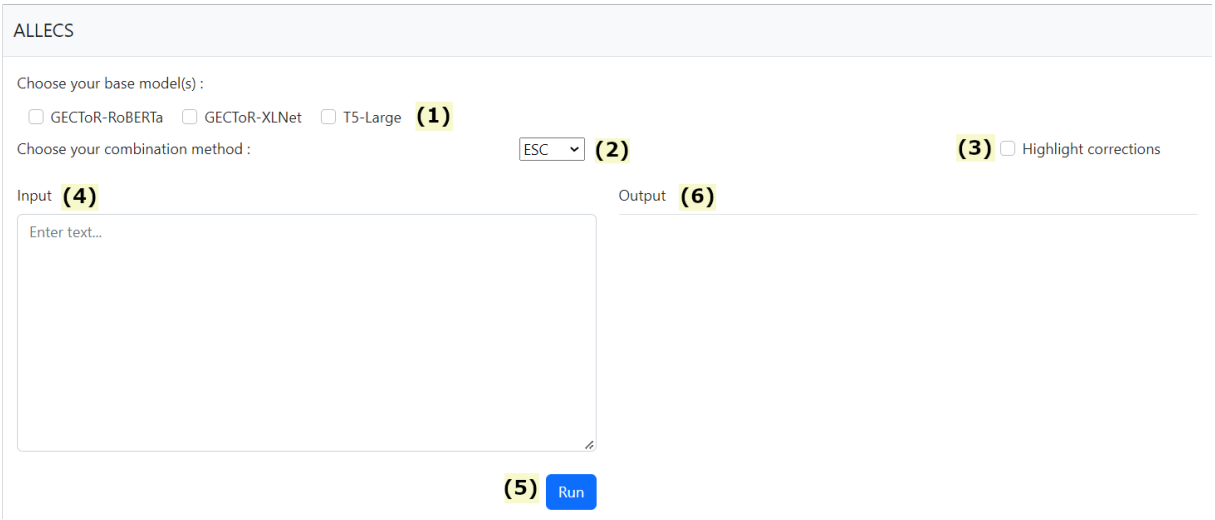

Figure 1: The user interface of ALLECS

a GEC system.

ALLECS can easily host GEC systems using different approaches, including system combination. Currently, ALLECS hosts two sequence-tagging base models, one sequence-to-sequence generation base model, and two combination methods. We briefly explain the base systems we use in ALLECS in Section 3. ALLECS also easily allows the addition of other base models. To the best of our knowledge, ALLECS is the first web application with a graphical user interface for GEC system combination methods.

2 System Overview

The interface of ALLECS consists of five components, which are base model selection, combination method selection, output mode, input text box, and output text box. The user interface of ALLECS is shown in Figure 1.

2.1 Base model selection

The user first needs to choose the base model(s). If the user chooses more than one base model, ALLECS will run a system combination method based on the combination method selected, as described in the next section. ALLECS includes three state-of-the-art GEC systems as the base models: GECToR Roberta (Omelianchuk et al., 2020), GECToR XLNet (Omelianchuk et al., 2020), and T5-Large (Rothe et al., 2021). We describe the base systems in more detail in Section 3.

2.2 Combination method selection

Next, the user needs to choose the combination method. If the user only chooses one base system, the selected combination method is ignored. ALLECS includes two state-of-the-art system combination methods, ESC (Qorib et al., 2022) and MEMT (Heafield and Lavie, 2010). We describe the system combination methods in more detail in Section 3.

2.3 Output mode

Users can choose to highlight the corrections by selecting the "Highlight corrections" box. If the user chooses to highlight the corrections, text spans in the output text that are different from the input text are highlighted in blue and a simple explanation of each correction can be displayed by clicking a highlighted text span. The appearance of highlighted corrections can be seen in Figure 2. Displaying corrections with simple explanations can help language learners to understand their mistakes better. We extracted the corrections with their edit types using ERRANT (Bryant et al., 2017).

2.4 Input text box

The user needs to put the text he wants to correct in the input text box and clicks the run button. The corrected text will then be displayed in the output text box. Most recent GEC base systems expect the input to be a single sentence tokenized with SpaCy (Honnibal et al., 2020) version 1.9, following the requirement from the BEA-2019 shared task. As such, an input text needs to be segmented into sentences and then tokenized with SpaCy before each

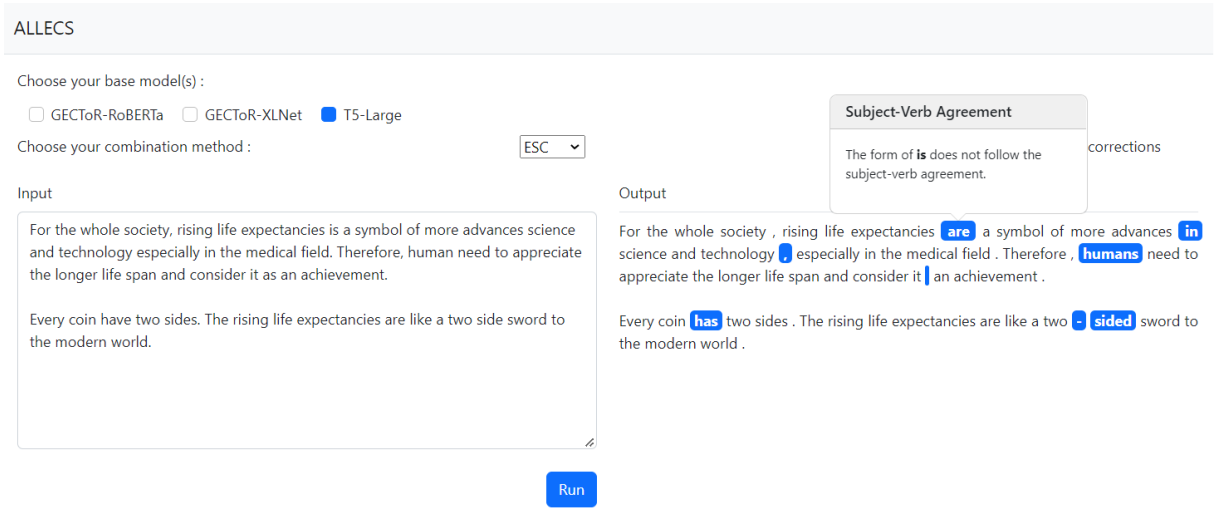

Figure 2: Text spans displayed in blue are the corrections made by the GEC system. A vertical blue bar without any text inside denotes text deletion. When a blue highlight or a blue bar is clicked, a simple explanation of the correction is shown.

sentence is given as the input to the GEC model. To retain the text structure, it is first split by line before segmented into sentences. This way, we can keep the information on which line a sentence should be printed. To segment a text into sentences, we follow the practice used in the NUCLE corpus (Dahlmeier et al., 2013) by using the nltk Punkt tokenizer (Bird and Loper, 2004; Kiss and Strunk, 2006).

2.5 Output text box

After a text is entered into the input text box and the "Run" button is clicked, the corrected text will appear in the output text box. As the base GEC systems are expected to work on tokenized input and output, the output text needs to be detokenized to look more natural. Since SpaCy does not have a detokenizer and the document context of the original input may no longer be relevant after a sentence is corrected, we use Moses (Koehn et al., 2007) to detokenize a sentence. We found that Moses can detokenize a sentence that is tokenized by SpaCy reasonably well, only missing some cases like the detokenization of "is n't" and "are n't" and removing spaces around hyphens. For these missed cases, we create simple rules to apply string replacement after Moses detokenization. Detokenization is not applied if the user chooses to highlight the corrections because the highlights need some room to make them clearly visible.

3 GEC Base Systems and Combination Methods

In ALLECS, we provide three base systems and two system combination methods. The base systems we provide come from two approaches, sequence-to-sequence generation (T5-Large) and sequence tagging (GECToR). The combination methods we provide also come from two approaches, edit-based (ESC) and text-based (MEMT) combination. The performance of the base systems and the combination methods on the BEA-2019 development set, CoNLL-2014 test set, and BEA-2019 test set is presented in Table 1. The scores of the base systems are presented in the top part of the table while the scores of the combination methods when combining the three base systems are presented in the bottom part.

3.1 T5-Large

T5 (Text-To-Text Transfer Transformer) (Raffel et al., 2020) is a large transformer model trained with a unified framework that converts all text-based language tasks into a text-to-text format. T5 is a sequence-to-sequence model with an architecture similar to the original Transformer (Vaswani et al., 2017).

Rothe et al. (2021) adapt T5 for grammatical error correction by fine-tuning the model on a new dataset they released, cLang-8. The cLang-8 dataset is made from re-labeling the Lang-8 dataset (Mizumoto et al., 2011; Tajiri et al., 2012), using a large model that is pre-trained with 50 billion doc-

Model	BEA-2019 Dev			CoNLL-2014			BEA-2019 Test		
	P	R	$F_{0.5}$	P	R	$F_{0.5}$	P	R	$F_{0.5}$
1. T5-Large	60.38	44.04	56.21	72.84	51.62	67.30	74.30	66.75	72.66
2. GECToR XLNet	66.00	34.14	55.62	77.49	40.15	65.34	79.20	53.90	72.40
3. GECToR Roberta	62.37	35.52	54.18	73.91	41.66	64.00	77.20	55.10	71.50
ESC	72.24	37.29	60.84	81.72	42.04	68.74	85.71	57.45	78.04
MEMT	61.82	44.02	57.19	70.64	50.20	65.32	75.41	66.44	73.42

Table 1: The top rows report the performance of the GEC systems that are provided as the base systems in ALLECS, while the bottom rows report the performance of the GEC system combination methods that are provided in ALLECS when combining the three base systems above.

uments from 101 languages and trained with the BEA-2019 training data. See (Rothe et al., 2021) for more details.

The T5 authors released the code[2] to train the GEC model but not their trained model. We use their original code to train the GEC base model using their hyper-parameter values[3].

3.2 GECToR

GECToR models GEC as a sequence tagging task by defining a set of token transformations. They defined two types of token transformations: basic transformations and g-transformations. The basic transformations include the keep, delete, and token-dependent append and replace transformations. The g-transformations are task-specific transformations such as merging two words, changing the verb form, changing the noun number, etc.

GECToR was built by fine-tuning a large pretrained model in three rounds of training. In the first round, they trained the model on 9M sentence pairs of synthetic data. In the last two rounds, the model is further trained on the BEA-2019 training data. At inference time, GECToR runs iteratively for a number of rounds. This helps to increase both precision and recall of the corrections. Despite running the inference multiple times, GECToR's inference speed is up to 10 times faster compared to models using the sequence-to-sequence approach. See (Omelianchuk et al., 2020) for more details.

In ALLECS, we use the XLNet and Roberta versions of GECToR, as the ensemble of these models produces the highest scores. We use the original source code and model weights[4] in ALLECS.

3.3 ESC

ESC is a system combination method that formulates the combination task as binary classification. ESC takes the union of all edits from the base systems and generates the features for each edit based on its edit type and inclusion in the base systems. ESC uses logistic regression to predict the probability that an edit is correct, and filters the edits based on a threshold and a greedy selection method.

At the time of writing, ESC is the highest-scoring GEC system on the CoNLL-2014 test set and the BEA-2019 test set, by combining T5-Large (Rothe et al., 2021), GECToR XLNet (Omelianchuk et al., 2020), GECToR Roberta (Omelianchuk et al., 2020), Riken & Tohoku (Kiyono et al., 2019), UEDIN-MS (Grundkiewicz et al., 2019), and Kakao&Bain (Choe et al., 2019). In ALLECS, for simplicity, we only provide the top three base systems since the performance of the ensemble of these three systems is still highly competitive with other state-of-the-art systems. We use ESC's original code[5] but slightly modify it to take inputs stored in memory rather than reading them from files. We then train the ensemble model for all possible base system configurations.

3.4 MEMT

MEMT is a system combination method that combines the base models' outputs by first aligning them and generating all possible candidate sentences based on the token alignment. Candidate generation has some constraints, such as no repetition, weak monotonicity, and completeness. For each candidate sentence, MEMT generates the features based on the language model score, n-gram similarity to each base model's output, and the sentence length. MEMT then learns the weights to score the features and uses the trained weights

[2] https://github.com/google-research/text-to-text-transfer-transformer/

[3] https://github.com/google-research-datasets/clang8/issues/3#issuecomment-913682092

[4] https://github.com/grammarly/gector/tree/fea1532608

[5] https://github.com/nusnlp/esc

to find the highest-scoring candidate sentence via beam search during inference.

MEMT was originally designed for combining machine translation models, but Susanto et al. (2014) have demonstrated that MEMT can effectively combine GEC models as well. In ALLECS, we use MEMT's original code[6] and train the ensemble model for all possible base system configurations.

4 System Design and Implementation

To make ALLECS more modular, we design it to have two parts: the web interface (front-end) and the base models' API (back-end). The web interface accepts a user's input text and handles the pre-processing and post-processing, while the base models' API focuses on generating a corrected sentence from the pre-processed input. This separation allows the base models' API to run on a GPU-powered server while the web interface runs on a CPU-focused server. However, both parts can also be run on the same server.

As Python has become the dominant programming language of choice for most NLP research projects, we develop ALLECS in Python. This makes ALLECS highly extensible such that it can be used to host new GEC systems in the future. The separation between the front-end and back-end parts also allows ALLECS to host other base models that are written in a different programming language or use different library dependencies. This also allows the web interface to work with other closed-source GEC models, as long as a GEC model provides an API to generate a corrected sentence.

We describe the process flow of ALLECS in Figure 3. All inputs are first split by line and segmented into sentences. The line index for each sentence is recorded to retain the text structure in the output. Then, the web interface tokenizes the sentences and combines them into mini-batches to be sent to the base models' API. If the user chooses to highlight the corrections or combine multiple base models with ESC, the web interface will also use ERRANT to parse the input sentences. After receiving the output sentences from each base model, the interface will then parse the base models' outputs using ERRANT if the user chooses to highlight corrections or use ESC. If not, the outputs are sent to MEMT if the user chooses to combine the models with MEMT. Otherwise, the output sentences are directly detokenized. Detokenization also applies to the combination method's output if the user selects more than one base model.

The correction speed of ALLECS is fast. Running on an NVIDIA Titan X GPU server with 12GB memory, GECToR Roberta can correct text at a speed of 723 words per second, GECToR XLNet at 640 words per second, and T5-Large at 37 words per second. Using ESC to combine base systems only adds a small amount of overhead. For example, using ESC to combine GECToR Roberta and T5-Large can correct text at a speed of 32 words per second, marginally slower than using T5-Large alone.

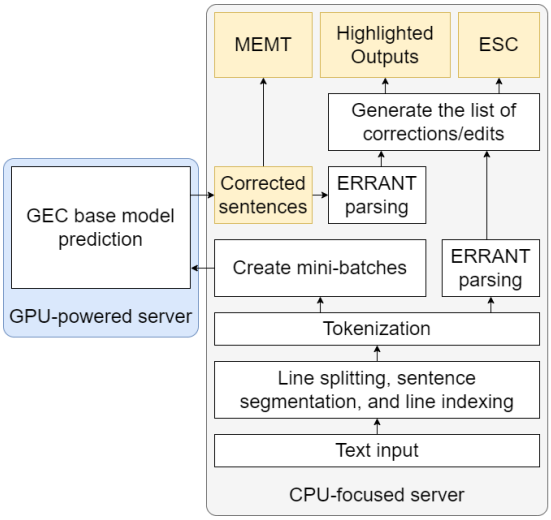

Figure 3: The process flow of ALLECS.

4.1 Web interface

We use flask version 2.0.3[7] and Bootstrap version 5[8] frameworks to develop the web interface. Bootstrap is a lightweight CSS and JavaScript framework that helps developers to design an interface with accessibility in mind, building a responsive layout and conforming with Web Content Accessibility Guideline (WCAG) 2.1. We design the web interface to only use colors with a contrast ratio above 4.5:1[9]. The web interface also works well when zoomed to 200%[10] and can be used with

[6] https://github.com/kpu/memt

[7] https://flask.palletsprojects.com/en/2.0.x/

[8] https://getbootstrap.com/docs/5.0/getting-started/introduction/

[9] Conforming to WCAG 2.1 minimum contrast recommendation https://w3.org/TR/WCAG/#contrast-minimum

[10] Conforming to WCAG 2.1 resize text recommendation https://www.w3.org/TR/WCAG/#resize-text

different screen dimensions and sizes[11]. With the responsive design, the position of the output box is moved to the bottom of the input box when opened from a screen with smaller size or with vertical orientation, such as on mobile phones, as shown in Figure 4.

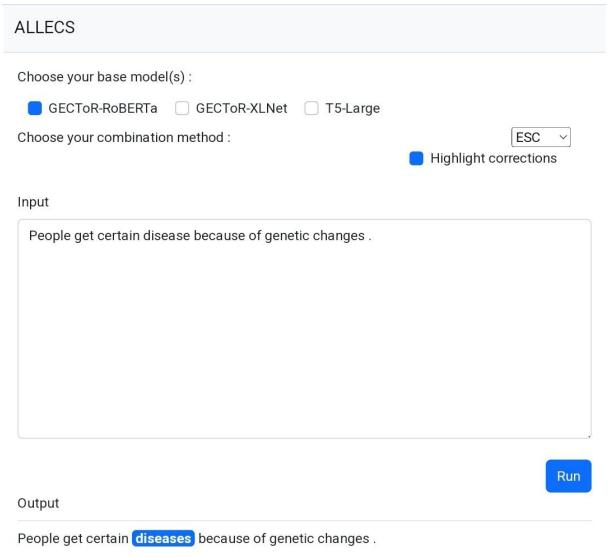

Figure 4: The interface layout on a vertical screen such as a mobile phone.

4.2 Base models' API

The base models' API hosts all base systems in ALLECS. The base models' API first loads each base system on different GPUs at start time. This approach makes the application persistently uses a large amount of GPU memory but allows much faster inference. We build the base models' API using flask version 2.0.3 and flask RESTful version 0.3.9[12] frameworks. The base models' API serves an API with the HTTP POST request method, expecting a JSON that contains a list of sentences on the keyword "text_input_list", and returns a JSON that contains a list of corrected sentences with the same length as the input list, on the keyword "text_output_list". Adding new base systems to ALLECS only requires changing a few lines of code.

5 Related Work

There are some ready-to-use web services for correcting English text such as services from Grammarly[13] and John Snow Labs[14], but those web services are not open-source. Thus, they are not customizable for deploying different GEC systems. In this section, we will discuss the comparison of ALLECS to other open-source English correction tools, namely GECko+ (Calò et al., 2021) and MiSS (Li et al., 2021).

5.1 GECko+

GECko+ (Calò et al., 2021) is a grammatical and discourse correction tool that combines a sentence-level GEC model, GECToR XLNet, and a sentence ordering model (Prabhumoye et al., 2020). When a user inputs a text into the system, it segments the text into sentences and corrects the sentences with GECToR before re-ordering them by the sentence ordering model.

Compared to ALLECS, GECko+ lacks the options of choosing the GEC base models and using system combination methods. It is also unclear how easy it is to extend GECko+ to other GEC systems. ALLECS does not include a sentence re-ordering model because it focuses on grammatical error correction, and re-ordering sentences can confuse the user and makes it harder for the user to learn from the corrections.

5.2 MiSS

MiSS (Li et al., 2021) is a comprehensive tool for machine translation that includes grammatical error correction as a feature. The main machine translation features of MiSS include: basic machine translation, simultaneous machine translation, and back-translation for quality evaluation. For the GEC part, it uses GECToR XLNet for English GEC and GECToR with BERT-chinese and BERT-japanese models for Chinese and Japanese GEC respectively.

Compared to ALLECS, MiSS also lacks the options of choosing the GEC base models and using system combination methods. It is also unclear how easy it is to extend MiSS to other GEC systems.

6 Conclusion

We have presented ALLECS, a web-based application for GEC that can be easily used by the general public. We design ALLECS to be accessible to as many users as possible, including users who have a slow Internet connection and who use mobile

[11] Conforming to WCAG 2.1 orientation recommendation https://www.w3.org/TR/WCAG/#orientation

[12] https://flask-restful.readthedocs.io/en/latest/

[13] https://www.grammarly.com/

[14] https://demo.johnsnowlabs.com/public/T5_LINGUISTIC/

phones as their main devices to connect to the Internet. In ALLECS, we provide three base GEC systems using two types of approaches, sequence-to-sequence generation and sequence tagging, as well as two GEC system combination methods from two types of approaches, edit-based and text-based combination. ALLECS is separated into two parts to make the application modular and easily extensible to other base GEC systems.

Limitations

ALLECS is currently limited to English GEC systems, but it can be extended to other languages by incorporating base GEC models for other languages and modifying the pre-processing and post-processing steps.

Acknowledgements

This research is supported by the National Research Foundation, Singapore under its AI Singapore Programme (AISG Award No: AISG-RP-2019-014). The computational work for this article was partially performed on resources of the National Supercomputing Centre, Singapore (https://www.nscc.sg).

References

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75.

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.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2022. Grammatical error correction: A survey of the state of the art. arXiv. https://arxiv.org/abs/2211.05166.

Eduardo Calò, Léo Jacqmin, Thibo Rosemplatt, Maxime Amblard, Miguel Couceiro, and Ajinkya Kulkarni. 2021. GECko+: a grammatical and discourse error correction tool. In *Actes de la 28e Conférence sur le Traitement Automatique des Langues Naturelles. Volume 3 : Démonstrations*, pages 8–11. ATALA.

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

Shamil Chollampatt, Duc Tam Hoang, and Hwee Tou Ng. 2016. Adapting grammatical error correction based on the native language of writers with neural network joint models. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1901–1911.

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.

Robert Dale and Adam Kilgarriff. 2011. Helping our own: The HOO 2011 pilot shared task. In *Proceedings of the 13th European Workshop on Natural Language Generation*, pages 242–249.

Anne Delaporte and Kalvin Bahia. 2020. The state of mobile internet connectivity 2022. Technical report, GSM Association.

David M. Eberhard, Gary F. Simons, and Charles D. Fennig. 2022. Ethnologue: Languages of the world.

Svetlana Glushkova, Denis Belotserkovich, Natalia Morgunova, and Yulia V. Yuzhakova. 2019. The role of smartphones and the internet in developing countries. *Revista ESPACIOS*, 40(27):10–18.

Roman Grundkiewicz, Marcin Junczys-Dowmunt, and Kenneth Heafield. 2019. Neural grammatical error correction systems with unsupervised pre-training on synthetic data. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 252–263.

Kenneth Heafield and Alon Lavie. 2010. CMU multi-engine machine translation for WMT 2010. In *Proceedings of the Joint Fifth Workshop on Statistical Machine Translation and MetricsMATR*, pages 301–306.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, and Adriane Boyd. 2020. spacy: Industrial-strength natural language processing in python.

Anisia Katinskaia and Roman Yangarber. 2021. Assessing grammatical correctness in language learning. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 135–146.

Tibor Kiss and Jan Strunk. 2006. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242.

Ola Knutsson, Teresa Cerrato Pargman, and Kerstin Severinson Eklundh. 2003. Transforming grammar checking technology into a learning environment for second language writing. In *Proceedings of the HLT-NAACL 03 Workshop on Building Educational Applications Using Natural Language Processing*, pages 38–45.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer, Ondřej Bojar, Alexandra Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation. In *Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics Companion Volume Proceedings of the Demo and Poster Sessions*, pages 177–180.

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. 2022. Type-driven multi-turn corrections for grammatical error correction. In *Findings of the Association for Computational Linguistics: ACL 2022*, pages 3225–3236.

Zuchao Li, Kevin Parnow, Masao Utiyama, Eiichiro Sumita, and Hai Zhao. 2021. MiSS: An assistant for multi-style simultaneous translation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 1–10.

Junwei Liao, Sefik Emre Eskimez, Liyang Lu, Yu Shi, Ming Gong, Linjun Shou, Hong Qu, and Michael Zeng. 2022. Improving readability for automatic speech recognition transcription. *ACM Transactions on Asian and Low-Resource Language Information Processing*.

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.

Maria Nadejde and Joel Tetreault. 2019. Personalizing grammatical error correction: Adaptation to proficiency level and L1. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 27–33.

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.

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.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170.

Shrimai Prabhumoye, Ruslan Salakhutdinov, and Alan W Black. 2020. Topological sort for sentence ordering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2783–2792.

Muhammad Reza Qorib, Seung-Hoon Na, and Hwee Tou Ng. 2022. Frustratingly easy system combination for grammatical error correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1964–1974.

Muhammad Reza Qorib and Hwee Tou Ng. 2022. Grammatical error correction: Are we there yet? In *Proceedings of the 29th International Conference on Computational Linguistics*, pages 2794–2800.

Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. 2020. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A simple recipe for multilingual grammatical error correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707.

Raymond Hendy Susanto, Peter Phandi, and Hwee Tou Ng. 2014. System combination for grammatical error correction. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 951–962.

Toshikazu Tajiri, Mamoru Komachi, and Yuji Matsumoto. 2012. Tense and aspect error correction for ESL learners using global context. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 198–202.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc.

Fan Yin, Quanyu Long, Tao Meng, and Kai-Wei Chang. 2020. On the robustness of language encoders against grammatical errors. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3386–3403.