LinggleWrite: a Coaching System for Essay Writing

Chung-Ting Tsai[1], **Jhih-Jie Chen**[2], **Ching-Yu Yang**[2], **Jason S. Chang**[2]
[1]Institute of Information Systems and Applications
[2]Department of Computer Science
National Tsing Hua University
{jjc, jason}@nlplab.cc

Abstract

This paper presents *LinggleWrite*, a writing coach that provides writing suggestions, assesses writing proficiency levels, detects grammatical errors, and offers corrective feedback in response to user's essay. The method involves extracting grammar patterns, training models for automated essay scoring (AES) and grammatical error detection (GED), and finally retrieving plausible corrections from a n-gram search engine. Experiments on public test sets indicate that both AES and GED models achieve state-of-the-art performance. These results show that *LinggleWrite* is potentially useful in helping learners improve their writing skills.

1 Introduction

Essay writing has been an essential part of language assessments (e.g., TOEFL, IELTS) but a challenging task for most students. To write a good essay not only requires sustained practice, but also demands instructional feedback from teachers. However, pressed with teaching load, teachers can only provide limited corrective feedback on students' essays. This has encouraged the development of computer-assisted writing systems to meet growing needs of automated feedback as a means of writing coaching. Computer Assisted Language Learning (CALL) has been an active field of computational linguistics and pedagogy. Some existing computer aided writing systems detect and correct grammatical errors, and give an overall score (e.g., *Grammarly* (www.grammarly.com) and *Pigai* (www.pigai.org)).

Instead of directly correcting users' essays, *Write&Improve* (writeandimprove.com) only marks highly-likely incorrect words on the grounds that automated grammatical error correction is still very imprecise. Recently, researchers have begun to apply neural network models to both automated essay scoring (AES) and grammatical error detection (GED), gaining significant improvement (e.g., Dong et al. (2017); Rei and Søgaard (2018)). However, these Web services fall short of providing sufficient "coaching" information (e.g., grammar patterns, collocations, examples) to learners to improve their writing skills.

Provide writing suggestions as a user types away or during editing is another emerging approach to coaching the learner. For example, *WriteAhead* (writeahead.nlpweb.org) provides context-sensitive suggestions, right in the process of writing or self-editing. Google recently released *Smart Compose* that offers users word or phrase completion suggestions while writing an email (Chen et al., 2019).

In line with these systems, we also suggest that feedback on learners' writings could be more effective if a system not only acts as an editor providing direct corrections, but also a coach performing grammatical error detection and offering interactive suggestions (Hearst, 2015). Moreover, illustrating word usage with bilingual examples can better help non-native English learners. This would enhance learners' skills of self-editing and pave the way to lifelong language learning.

With that in mind, we developed a web-based system *LinggleWrite* (f.linggle.com) with many assistive writing functions. With *LinggleWrite* users can write or paste their essays and get informative feedback including just-in-time writing suggestions, essay scoring, error detection, and related word usage information retrieved from Linggle(linggle.com).

2 The *LinggleWrite* System

The system consists of 4 components: (1) Interactive Writing Suggestion, (2) Essay Scoring, (3) Grammatical Error Detection, and (4) Corrective

Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pages 127–133
July 5 - July 10, 2020.

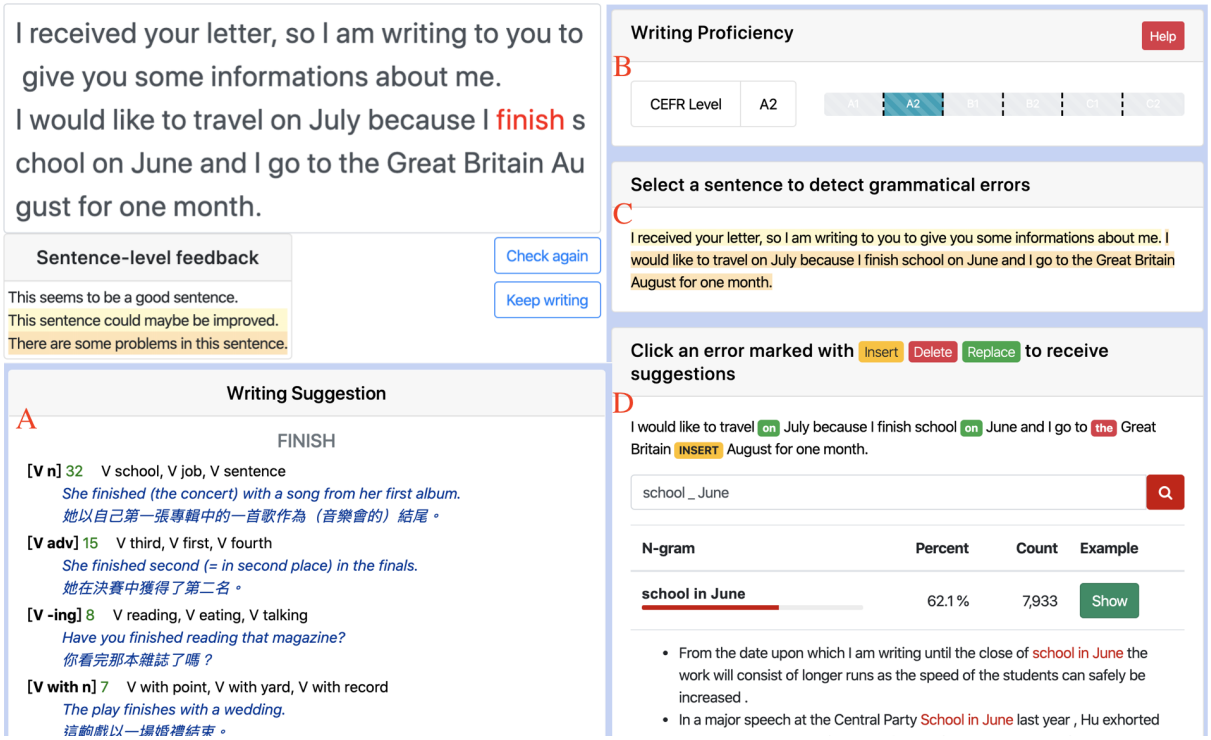

Figure 1: The screenshot of the system *LinggleWrite*

Feedback. The first component, Writing suggestion, will help users with word usage information while writing. The other three components are aimed at providing evaluation and constructive feedback after a user finishes writing. The system is available at `f.linggle.com`. We'll describe each component as follows.

2.1 Interactive Writing Suggestion

When a user begins to write an essay, the system responds with prompts of related grammar patterns, collocations, and bilingual examples. These continuous writing suggestions are based on the last word or phrase the user has entered. Additionally, the user can get information of a certain word by mousing over it. For example, suggestions for "finish" are shown in Section A of Figure 1 (bottom left). Once finishing the writings, the user can click the *Check* button triggering the following components.

2.2 Essay Scoring

After accepting an essay longer than 30 words, *LinggleWrite* assesses user's writing proficiency. The assessment is provided in the form of CEFR Levels[1] (A1-C2) as shown in Section B of Figure 1 (top right).

[1]https://www.coe.int/en/web/common-european-framework-reference-languages/level-descriptions

2.3 Grammatical Error Detection

LinggleWrite tries to detect potential grammatical errors in each sentence. Sentences with potential errors are marked with yellow (1 possible error) or orange (2 or more possible errors) background, as shown in Section C of Figure 1 (center right). The user can click on an erroneous sentence to demand GED results. *LinggleWrite* marks suspicious words with orange, red or green, suggesting to insert a word, delete the word, or replace the word respectively, as shown in Section C of Figure 1 (center right). Subsequently, the user can click on an error to display plausible corrective suggestions returned by a n-gram search engine.

2.4 Corrective Feedback

We present corrective suggestions according to the context and the edit type (i.e., insertion, deletion, replacement), using an existing linguistic search engine, *Linggle* (Boisson et al., 2013). An example of corrective suggestions for the sentence *"I finished school on June"* is shown in Section E in Figure 1 (bottom right). *LinggleWrite* detects "on" probably requiring a replacement edit. We convert the detected error into a Linggle query to search for more appropriate expressions, and provide the user with the search result "school in June' for

considerations.

3 Method

To develop *LinggleWrite*, we extract the most common grammar patterns from a corpus in order to provide writing suggestions. Additionally, we develop models for AES and GED based on annotated learner corpora. We retrieve corrective feedback by querying a linguistic search engine according to the predicted edit type of an error. We describe the process in detail in the following subsections.

3.1 Extracting Grammar Patterns

We extract grammatical patterns, collocations and bilingual examples for keywords from a given corpus to provide writing suggestions in the interactive writing session. Our extraction process includes four steps.

In Step (1), we build a dictionary of grammar patterns of verbs, nouns and adjectives based on Francis et al. (1996). For example, the grammar patterns of the word *play* are **V n**, **V n in n**, etc.

In Step (2), we parse sentences from Corpus of Contemporary American English (COCA) and Cambridge online dictionary (CAM) using a dependency parser to extract grammar patterns and collocations based on the templates in Step (1). For example, the extracted grammar pattern and collocation from the sentence "Schools **play** an important role **in** society" are "**V n in n**" and "**society**".

In Step (3), for each keyword, we count and filter out patterns and collocations based on mean and standard deviation. Finally, we use GDEX method (Kilgarriff et al., 2008) to select the best monolingual and bilingual examples from COCA and CAM for each pattern.

3.2 Scoring an Essay

We formulate AES as a regression problem and train a neural model for this task. We investigate two neural network architectures with different input formats: word-based models and sentence-based models, which learn essay representation based on word sequences and sentence sequences respectively. We build our word-based models upon CNN, LSTM and Bi-LSTM (Taghipour and Ng, 2016), while sentence-level models upon the LSTM-LSTM and LSTM-CNN framework (Dong et al., 2017). Moreover, we further extend both sentence-based models and word-based models by adding the attention mechanism after the neural layer, attempting to select the sentences or words to focus on for effective scoring. Our models are similar to other sentence-based and word-based neural AES model (e.g., Taghipour and Ng (2016); Dong et al. (2017)), but we use a different training set, EFCAMDAT (Geertzen et al., 2013) and output format, CEFR levels, to train our model.

3.3 Detecting Grammatical Errors

We formulate GED as a sequence labeling problem and develop a neural sequence labeling model to deal with the problem.

An existing GED method proposed by Rei and Yannakoudakis (2016) takes tokens as input and predicts whether each token is correct in the sentence as output. We extend their model by changing the binary error tag schema (*Incorrect and Correct*) into a more informative DIRC tag schema (*Delete, Insert, Replace, and Correct*), with the goal of providing learners more specific suggestions (i.e., the edit type of an error) to revise their essay. We train a GED model based on Bi-LSTM with a Conditional Random Field layer (CRF). To improve the GED model, we add Bidirectional Encoder Representations from Transformers (BERT), which significantly outperforms other embedding schemes in many tasks (Devlin et al., 2018). In addition, we also add a character-based word embedding, Flair, which captures more contextual information (Akbik et al., 2018). Our training process is divided into two steps.

In Step (1), we convert sentences with error annotations into unedited sentences and DIRC tags (i.e., <[-,-]> for *Delete*, tokens preceded by <{+,+}> for *Insert*, <[-,-]{+,+}> for *Replace* and tokens with no edit tag for *Correct*). For example, the sentence "I believe there are {+a+} lot of [-why-]{+ways+} enjoy [-the-] shopping." is converted to "I believe there are lot of why enjoy the shopping ." and "<C C C C I C R C D C C>". These two sequences are treated as the input and output of a neural GED model respectively. Note that the token to be inserted ({+*a*+}) is not in the unedited sentence, and the right token *lot* is labeled *I* instead.

In Step (2), we train a neural GED model for a grammatical error detector using a BiLSTM-CRF architecture. We first combine BERT embeddings (Devlin et al., 2018) with Flair embeddings (Akbik et al., 2018) to form word embeddings and then encode each token in a given sentence into a fixed-length vector. Finally, these embeddings are fed

Operators	Corresponding edit types	Description	Example
*	Insertion Edit	match zero or any words	good * this
_	Replacement Edit	match one word	not _ me to
?	Deletion Edit	search for TERM optionally	discuss ?about this issue

Table 1: Query operator instruction

into BiLSTM-CRF network to compute and output a DIRC label sequence.

3.4 Retrieving Suggestions for Detected Errors

To retrieve writing suggestions for detected errors, we design queries for each edit type to search for more plausible corrections using *Linggle*, a linguistic search engine on a web-based dataset of one trillion words (Boisson et al., 2013).

Linggle has different query functions and operators to search word usage in context as shown in Figure 1. These query functions enable the system to query zero, one or multiple words. For example, "play * role" is intended to search for a maximum span of three intervening words. We use three operators ("?", "*", "_") to retrieve corrective suggestions for the three edit types, as described below.

Deletion edit: We use the "?" operator before a word tagged with "D" to search for n-grams with or without the word in question. For example, receiving the sentence *"We discuss **about** this issue."* as input, our GED model outputs the sequence "C C **D** C C C". Then, we generate the query "discuss **?about** this issue" to search *Linggle* for corrective suggestions.

Insertion edit: We use the "*" operator before a word tagged with "I" to search for ngrams with additional words around this word. For example, an insertion edit on "this" is detected in the sentence *"I am good **this** sport."* (the GED model output "C C C **I** C"), and thus a *Linggle* query are formulated as "good ***** this".

Replacement edit: A word tagged with "R" indicates replacement required. We first check if the word is misspelled using *enchant*[2] library. If misspelled, we replace the word with candidates by *enchant* (e.g., 'moey' → 'money/mopey/mosey'). If not, we use the "_" operator to search for alternative n-grams. For example, the GED output of the sentence *"The driver did not **accept** me to get on the bus."* would be "C C C C **R** C C C C C C C C". Thus, we use the query "not _ me to" to search for replacement.

[2]https://github.com/AbiWord/enchant

4 Experiments

4.1 Datasets

We used the EF-Cambridge Open Language Database (EFCAMDAT) (Geertzen et al., 2013) to train our AES model. This dataset contains about 1.2 million essays with over 83 million words written by approximately 174,000 learners with a wide range of CEFR levels (A1-C2) (language proficiency level). We used the student essays as input and the CEFR level assigned by a grader as output to train the AES model. Due to the imbalanced distribution of levels as shown in Table 2, we randomly selected 1,903 essays from each level and then used 5-fold cross validation for training and evaluation.

CEFR Level	#Essays	#Training
A1	460,614	1,903
A2	300,188	1,903
B1	166,453	1,903
B2	60,844	1,903
C1	14,551	1,903
C2	1,903	1,903

Table 2: Description of the EFCAMDAT dataset

To train the GED model, we use the First Certificate in English dataset (FCE). This dataset contains 1,224 essays written by English learners who took the First Certificate in English (FCE) exam. These essays have been manually tagged based on 77 error types (Yannakoudakis et al., 2011). We used 30,953 sentences from FCE for training, 2,720 for testing, and 2,222 for development. We followed the approach of Rei and Yannakoudakis (2016) in our experiment, but converted the dataset into DIRC format as described in Section 3.3.

Model	Binary Task			DIRC Task								
	Incorrect tag			Insertion tag			Replacement tag			Deletion tag		
	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$	Prec.	Rec.	$F_{0.5}$
Rei and Søgaard (2018)	65.5	28.6	52									
BiLSTM-CRF + word2vec	**89**	13.8	42.6	**57.2**	12.1	32.9	**82.9**	22.4	53.9	**67.6**	3.1	13.2
BiLSTM-CRF + Flair	68.9	24.6	50.7	53.8	20.2	40.4	72.8	28.3	55.4	59.6	10.1	30.17
BiLSTM-CRF + BERT	71.1	35.7	59.4	53.2	23.8	42.7	73.1	36.1	60.7	53.9	24.1	43.2
BiLSTM-CRF + BERT +Flair	72.3	**36.7**	**60.6**	54.6	**25.3**	**44.3**	73.5	**40.6**	**63.3**	59	**24.9**	**46.3**

Table 3: Evaluation on FCE-public test set in DIRC task and binary task

4.2 Hyperparameters

For the AES model, we optimized the trained model using RMSProp (Dauphin et al., 2015) optimizer with learning rate 0.001 and the maximum gradient norm was set to 0.9. We used pre-trained 100-dimensional GloVe vectors (Pennington et al., 2014) as input. The hidden layer size of LSTM and Bi-LSTM was set to 100. For CNN models, we used a window size of 5 and hidden layer size of 100. We applied dropout on the neural network layer to avoid overfitting, with dropout probabilities set to 0.2. The batch size was 32 and each model was trained for 50 epochs.

For the GED model, we set parameters different from previous work (Rei and Yannakoudakis, 2016). We use the publicly available pre-trained word embeddings GoogleNews word vectors (word2vec) (Mikolov et al., 2013), Flair (Akbik et al., 2018), and BERT[3] (Devlin et al., 2018) to represent words. Flair embeddings were trained on the 1-billion word corpus used in Chelba et al. (2013) and the embedding size (both forward and backward) was 2048. As for BERT, we utilized *bert-base-uncased* model which is trained on the English Wikipedia (2.5G words) and *BooksCorpus* (0.8G words). We employed 2-layer Bi-LSTM with CRF to develop for GED model and set the hidden layer size of Bi-LSTM to 256. We used SGD optimizer with learning rate 0.01, with maximum gradient norm set to 1. We applied dropout on both embedding and Bi-LSTM layers with dropout probabilities 0.5. We trained the network for 150 epochs and selected the best model with the highest F1 score on the development set.

5 Evaluation

For the AES task, we adopted quadratic weighted Kappa (QWK) as our evaluation metric, which was used in Automated Student Assessment Prize (ASAP) competition and several AES researches (Taghipour and Ng, 2016; Vaswani et al., 2017; Dong et al., 2017). For the GED task, we follow the previous research by Rei and Yannakoudakis (2016) and use precision, recall and $F_{0.5}$ to evaluate our GED model.

Table 3 presents the results of different GED models on the FCE testset with binary and DIRC format to compare our results with the state-of-the-art method proposed by Rei and Søgaard (2018) using the binary schema. Table 3 shows that **BiLSTM-CRF+BERT+Flair** performs substantially better than the other GED models and achieve state-of-the-art performance on the FCE test set. Interestingly, we note that the model with word2vec pre-trained word embeddings achieves the highest precision but the lowest recall. As for the DIRC schema, **BiLSTM-CRF+BERT+Flair** performs the best among all models. Importantly, the DIRC model performs comparably to the binary model while providing more informative feedback (i.e., the edit type) for learners to self-edit their essays. It is also worth noting that for GED and GEC tasks multiple answers are acceptable and there is low inter-annotator agreement (Rozovskaya and Roth, 2010). Bryant and Ng (2015) pointed out even human annotators can only achieve 72.8 $F_{0.5}$ score at the best against the gold standard annotations of multiple annotators in GEC tasks. Thus, it is fair to say that the performance of our model against one gold standard annotation are underestimated and not far from human annotators, thus acceptable for an application.

Table 4 shows results of different network architectures on the AES task. As we can see in Table 4, **LSTN-LSTM-ATT** achieves the best performance among all models. In addition, we find that sentence-level models perform better than word-level ones in general. Furthermore, we also observe that the model with attention mechanism per-

[3]https://github.com/google-research/bert#pre-trained-models

Model	Model Type	Avg. QWK score
CNN	Word-level	0.902
LSTM	Word-level	0.927
Bi-LSTM	Word-level	0.921
LSTM + attention	Word-level	0.931
CNN-CNN	Sentence-level	0.934
LSTM-LSTM	Sentence-level	0.937
CNN-LSTM-ATT	Sentence-level	0.952
LSTM-LSTM-ATT	Sentence-level	**0.957**

Table 4: Average QWK scores on EFCAMDAT

forms slightly better than the other without attention mechanism. Besides, the result (i.e., QWK score 0.957) shows our neural models are efficient to predict scores in EFCAMDAT, comparing with other datasets as Automated Student Assessment Prize[4] (ASAP). Trained on ASAP, the character-based model with CNN-LSTM proposed by Taghipour and Ng (2016) scores QWK 0.761, and the sentence-based model with LSTM-CNN-att proposed by Taghipour and Ng (2016) achieves QWK score 0.764.

6 Conclusion and Future Work

In summary, we have presented an writing environment that supports interactive writing suggestions, scoring, error detection and corrective feedback. For the interactive writing task, we provide grammatical suggestions, collocations, and bilingual examples, to guide the user towards writing fluently. For the GED task, we proposed a new label schema, DIRC. Experiments show that the proposed label schema achieves comparable performance (on binary task) while providing more informative feedback. In addition, we leverage an existing linguistic search engine to provide corrective suggestions for each error type.

Many avenues exist for future research and improvement of our system. For example, the method for introducing additional training data or generating artificial training data could be implemented to improve the performance. An interesting direction to explore is re-ranking corrective suggestions, so that the suggestion more relevant to the original sentence goes to the top. Yet another direction of research would be to detect fine-grained error types. Finally, our system currently providing additional Chinese translations for English examples. Obviously we could easily provide languages translations by changing a bilingual dictionary.

[4]https://www.kaggle.com/c/asap-aes

Acknowledgment

This work is supported by Ministry of Science and Technology, Taiwan under Grant No. 109-2639-M-007-001-, No. 109-2634-F-001-010-, and National Tsing Hua University under Grant No. 109Q2729E1.

References

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 1638–1649. Association for Computational Linguistics.

Joanne Boisson, Ting-Hui Kao, Jian-Cheng Wu, Tzu-Hsi Yen, and Jason S Chang. 2013. Linggle: a web-scale linguistic search engine for words in context. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 139–144.

Christopher Bryant and Hwee Tou Ng. 2015. How far are we from fully automatic high quality grammatical error correction? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 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. Technical report, Google.

Mia Xu Chen, Benjamin N Lee, Gagan Bansal, Yuan Cao, Shuyuan Zhang, Justin Lu, Jackie Tsay, Yinan Wang, Andrew M Dai, Zhifeng Chen, et al. 2019. Gmail smart compose: Real-time assisted writing. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 2287–2295.

Yann N. Dauphin, Harm de Vries, and Yoshua Bengio. 2015. Equilibrated adaptive learning rates for non-convex optimization. In *Proceedings of the 28th International Conference on Neural Information Processing Systems - Volume 1*, NIPS'15, pages 1504–1512, Cambridge, MA, USA. MIT Press.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*.

Fei Dong, Yue Zhang, and Jie Yang. 2017. Attention-based recurrent convolutional neural network for automatic essay scoring. In *Proceedings of the 21st*

Conference on Computational Natural Language Learning (CoNLL 2017), pages 153–162. Association for Computational Linguistics.

Gill Francis, Susan Hunston, and Elizabeth Manning. 1996. Grammar patterns 1: verbs. *NY: HarperCollins Publication*.

Jeroen Geertzen, Theodora Alexopoulou, and Anna Korhonen. 2013. Automatic linguistic annotation of large scale l2 databases: The ef-cambridge open language database. In *Proceedings of SLRF 2012*.

Marti A Hearst. 2015. Can natural language processing become natural language coaching? In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1245–1252.

Adam Kilgarriff, Milos Husák, Katy McAdam, Michael Rundell, and Pavel Rychlỳ. 2008. Gdex: Automatically finding good dictionary examples in a corpus. In *Proc. Euralex*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119, USA. Curran Associates Inc.

Jeffrey Pennington, Richard Socher, and Christopher D. Manning. 2014. Glove: Global vectors for word representation. In *In EMNLP*.

Marek Rei and Anders Søgaard. 2018. Jointly learning to label sentences and tokens. *CoRR*, abs/1811.05949.

Marek Rei and Helen Yannakoudakis. 2016. Compositional sequence labeling models for error detection in learner writing. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1181–1191. Association for Computational Linguistics.

Alla Rozovskaya and Dan Roth. 2010. Annotating esl errors: Challenges and rewards. In *Proceedings of the NAACL HLT 2010 fifth workshop on innovative use of NLP for building educational applications*, pages 28–36. Association for Computational Linguistics.

Kaveh Taghipour and Hwee Tou Ng. 2016. A neural approach to automated essay scoring. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1882–1891. Association for Computational Linguistics.

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.