AACL-IJCNLP 2020

Natural Language Processing Techniques for Educational Applications

Proceedings of the Sixth Workshop

December, 2020
Suzhou, China

Order copies of this and other ACL proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-952148-99-6

The proceedings and workshop is sponsored by
State Language Commission Research Fund of China(YB135-90).

Preface

Welcome to the 6th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA 2020), with a Shared Task on Chinese Grammatical Error Diagnosis.

The development of Natural Language Processing (NLP) has advanced to a level that affects the research landscape of many academic domains and has practical applications in many industrial sectors. On the other hand, educational environment has also been changed deeply, due to the significant impact from the international society and emergency disasters like CoVID19. With the impact, this workshop focuses on the NLP techniques applied to the educational environment. Research issues in this direction have gained more and more attention.

This is the 6th workshop held in the Asian area, with the first one NLPTEA 2014 workshop being held in conjunction with the 22nd International Conference on Computer in Education (ICCE 2014) from Nov. 30 to Dec. 4, 2014 in Japan. The second edition NLPTEA 2015 workshop was held in conjunction with the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (ACL-IJCNLP 2015) from July 26- 31 in Beijing, China. The third version NLPTEA 2016 workshop was held in conjunction with the 26th International Conference on Computational Linguistics (COLING 2016) from December 11- 16 in Osaka, Japan. The fourth edition NLPTEA 2017 workshop was held in conjunction with the 8th International Joint Conference on Natural Language Processing (IJCNLP 2017) from November 27- December 1 in Taipei, China. The fifth NLPTEA 2018 workshop was colocated with 56th Annual Meeting of the Association for Computational Linguistics from July 15- 19 in Melbourne, Australia. This year, we continue to promote this research line by holding the workshop in conjunction with the first Asia Chapter of ACL conference and also holding the fifth shared task on Chinese Grammatical Error Diagnosis.

In this year, we received 28 valid submissions for research issues, each of which was reviewed by at least two experts, and had 17 teams participating in the shared task, submitting their task reports. In total, there are 11 oral papers and 7 posters accepted. We also organize a keynote speech session in this workshop. The invited speakers, Professor Yuming Li and Professor Jihua Song are expected to deliver great talks on language education across human and machine and Chinese intelligent education online.

We would like to thank the program committee members for their hard work in completing the review tasks. Their collective efforts achieved quality reviews of the submissions within a few weeks. Great thanks should also go to the speakers, authors, and participants for the tremendous supports in making the workshop a success.

It's a great pity that we cannot meet at Suzhou city as planned, due to the impact of CoVID19. Still we welcome you to the first online NLPTEA, and wish you enjoy the workshop.

NLPTEA 2020 Workshop Chairs
Erhong Yang, Beijing Advanced Innovation Center for Language Resources
Endong Xun, Institute of Language Intelligence
Baolin Zhang, Research Institute of Chinese International Education
Gaoqi Rao, Research Institute of Chinese International Education
Beijing Language and Culture University

Workshop Organizers:

Erhong Yang, Beijing Advanced Innovation Center for Language Resources
Endong Xun, Institute of Language Intelligence
Baolin Zhang, Research Institute of Chinese International Education
Gaoqi Rao, Research Institute of Chinese International Education
Beijing Language and Culture University

Program Committee:

Ting Liu, Harbin Institute of Technology
Miao Fan, Baidu Co., Ltd.
Ruiji Fu, iFlyTek Co., Ltd.
Tingting He, Center China Normal University
Po Hu, Center China Normal University
Renfen Hu, Beijing Normal University
Lung-hao Lee, National Center University
Bin Li, Nanjing Normal University
Changliang Li, Kingsoft Co., Ltd.
Zhenghua Li, Soochow University
Pengyuan Liu, Beijing Language and Culture University
Dawei Lu, Renmin University of China
Shan Wang, University of Macau
Wenxin Xiong, Beijing Foreign Studies University
Chi Xiu, Microsoft (China) Co., Ltd.
Dong Yu, Beijing Language and Culture University
Liang-Chih Yu, Yuan-Ze University
Tianlin Yang, Beijing Language and Culture University
Min Zhang, Soochow University
Yue Zhang, Westlake University

Additional Reviewers:

Chengwen Wang, Beijing Language and Culture University

Invited Speaker:

Yuming Li, Beijing Language and Culture University
Jihua Song, Beijing Normal University

Table of Contents

Non-Autoregressive Grammatical Error Correction Toward a Writing Support System
Hiroki Homma and Mamoru Komachi 1

Arabisc: Context-Sensitive Neural Spelling Checker
Yasmin Moslem, Rejwanul Haque and Andy Way 11

LXPER Index 2.0: Improving Text Readability Assessment Model for L2 English Students in Korea
Bruce W Lee and Jason Lee 20

Overview of NLPTEA-2020 Shared Task for Chinese Grammatical Error Diagnosis
Gaoqi Rao, Erhong Yang and Baolin Zhang 25

Combining ResNet and Transformer for Chinese Grammatical Error Diagnosis
Shaolei Wang, Baoxin Wang, Jiefu Gong, Zhongyuan Wang, Xiao Hu, Xingyi Duan, Zizhuo Shen, Gang Yue, Ruiji Fu, Dayong Wu, Wanxiang Che, Shijin Wang, Guoping Hu and Ting Liu 36

Chinese Grammatical Error Diagnosis with Graph Convolution Network and Multi-task Learning
Yikang Luo, Zuyi Bao, Chen Li and Rui Wang 44

Integrating BERT and Score-based Feature Gates for Chinese Grammatical Error Diagnosis
Yongchang Cao, Liang He, Robert Ridley and Xinyu Dai 49

BERT Enhanced Neural Machine Translation and Sequence Tagging Model for Chinese Grammatical Error Diagnosis
Deng Liang, Chen Zheng, Lei Guo, Xin Cui, Xiuzhang Xiong, Hengqiao Rong and Jinpeng Dong 57

A Hybrid System for NLPTEA-2020 CGED Shared Task
Meiyuan Fang, Kai Fu, Jiping Wang, Yang Liu, Jin Huang and Yitao Duan 67

Chinese Grammatical Error Correction Based on Hybrid Models with Data Augmentation
Yi Wang, Ruibin Yuan, Yan'gen Luo, Yufang Qin, NianYong Zhu, Peng Cheng and Lihuan Wang 78

TMU-NLP System Using BERT-based Pre-trained Model to the NLP-TEA CGED Shared Task 2020
Hongfei Wang and Mamoru Komachi 87

CYUT Team Chinese Grammatical Error Diagnosis System Report in NLPTEA-2020 CGED Shared Task
Shih-Hung Wu and Junwei Wang 91

Chinese Grammatical Error Diagnosis Based on RoBERTa-BiLSTM-CRF Model
Yingjie Han, Yingjie Yan, Yangchao Han, Rui Chao and Hongying Zan 97

Chinese Grammatical Errors Diagnosis System Based on BERT at NLPTEA-2020 CGED Shared Task
Hongying Zan, Yangchao Han, Haotian Huang, Yingjie Yan, Yuke Wang and Yingjie Han 102

Chinese Grammatical Error Detection Based on BERT Model
Yong Cheng and Mofan Duan 108

Named-Entity Based Sentiment Analysis of Nepali News Media Texts
Birat Bade Shrestha and Bal Krishna Bal 114

SEMA: Text Simplification Evaluation through Semantic Alignment
Xuan Zhang, Huizhou Zhao, KeXin Zhang and Yiyang Zhang 121

A Corpus Linguistic Perspective on the Appropriateness of Pop Songs for Teaching Chinese as a Second Language
Xiangyu Chi and Gaoqi Rao 129

Conference Program

Friday, December 4, 2020

9:00–9:10 ***Opening Remarks***

9:10–10:50 **Keynotes Session**

9:10–10:00 *Keynote: Vision and Impact of Intelligent Essay Scoring*
Yuming Li

10:00–10:50 *Keynote: Intelligent Techniques and International Chinese Education Online*
Jihua Song

10:50–11:00 ***Coffe Break***

11:00–12:00 **Oral Seesion1: NLP Techniques for Education**

11:00–11:15 *Non-Autoregressive Grammatical Error Correction Toward a Writing Support System*
Hiroki Homma and Mamoru Komachi

11:15–11:30 *Arabisc: Context-Sensitive Neural Spelling Checker*
Yasmin Moslem, Rejwanul Haque and Andy Way

11:30–11:45 *LXPER Index 2.0: Improving Text Readability Assessment Model for L2 English Students in Korea*
Bruce W Lee and Jason Lee

11:45–12:00 *Overview of NLPTEA-2020 Shared Task for Chinese Grammatical Error Diagnosis*
Gaoqi Rao, Erhong Yang and Baolin Zhang

12:00–14:00 ***Lunch Break***

Friday, December 4, 2020 (continued)

14:00–16:20 Oral Session2: Chinese Grammatical Error Diagnosis

14:00–14:20 *Combining ResNet and Transformer for Chinese Grammatical Error Diagnosis*
Shaolei Wang, Baoxin Wang, Jiefu Gong, Zhongyuan Wang, Xiao Hu, Xingyi Duan, Zizhuo Shen, Gang Yue, Ruiji Fu, Dayong Wu, Wanxiang Che, Shijin Wang, Guoping Hu and Ting Liu

14:20–14:40 *Chinese Grammatical Error Diagnosis with Graph Convolution Network and Multi-task Learning*
Yikang Luo, Zuyi Bao, Chen Li and Rui Wang

14:40–15:00 *Integrating BERT and Score-based Feature Gates for Chinese Grammatical Error Diagnosis*
Yongchang Cao, Liang He, Robert Ridley and Xinyu Dai

15:00–15:20 *BERT Enhanced Neural Machine Translation and Sequence Tagging Model for Chinese Grammatical Error Diagnosis*
Deng Liang, Chen Zheng, Lei Guo, Xin Cui, Xiuzhang Xiong, Hengqiao Rong and Jinpeng Dong

15:20–15:40 *A Hybrid System for NLPTEA-2020 CGED Shared Task*
Meiyuan Fang, Kai Fu, Jiping Wang, Yang Liu, Jin Huang and Yitao Duan

15:40–16:00 *Chinese Grammatical Error Correction Based on Hybrid Models with Data Augmentation*
Yi Wang, Ruibin Yuan, Yan‘gen Luo, Yufang Qin, NianYong Zhu, Peng Cheng and Lihuan Wang

16:00–16:20 *TMU-NLP System Using BERT-based Pre-trained Model to the NLP-TEA CGED Shared Task 2020*
Hongfei Wang and Mamoru Komachi

Friday, December 4, 2020 (continued)

16:20–17:45 Poster Session

16:20–16:32 *CYUT Team Chinese Grammatical Error Diagnosis System Report in NLPTEA-2020 CGED Shared Task*
Shih-Hung Wu and Junwei Wang

16:32–16:44 *Chinese Grammatical Error Diagnosis Based on RoBERTa-BiLSTM-CRF Model*
Yingjie Han, Yingjie Yan, Yangchao Han, Rui Chao and Hongying Zan

16:44–16:56 *Chinese Grammatical Errors Diagnosis System Based on BERT at NLPTEA-2020 CGED Shared Task*
Hongying Zan, Yangchao Han, Haotian Huang, Yingjie Yan, Yuke Wang and Yingjie Han

16:56–17:08 *Chinese Grammatical Error Detection Based on BERT Model*
Yong Cheng and Mofan Duan

17:08–17:20 *Named-Entity Based Sentiment Analysis of Nepali News Media Texts*
Birat Bade Shrestha and Bal Krishna Bal

17:20–17:32 *SEMA: Text Simplification Evaluation through Semantic Alignment*
Xuan Zhang, Huizhou Zhao, KeXin Zhang and Yiyang Zhang

17:32–17:44 *A Corpus Linguistic Perspective on the Appropriateness of Pop Songs for Teaching Chinese as a Second Language*
Xiangyu Chi and Gaoqi Rao

Friday, December 4, 2020 (continued)

17:45–17:55 Closing Remark

Non-Autoregressive Grammatical Error Correction Toward a Writing Support System

Hiroki Homma and **Mamoru Komachi**
Tokyo Metropolitan University
homma-hiroki@ed.tmu.ac.jp komachi@ed.tmu.ac.jp

Abstract

There are several problems in applying grammatical error correction (GEC) to a writing support system. One of them is the handling of sentences in the middle of the input. Till date, the performance of GEC for incomplete sentences is not well-known. Hence, we analyze the performance of each model for incomplete sentences. Another problem is the correction speed. When the speed is slow, the usability of the system is limited, and the user experience is degraded. Therefore, in this study, we also focus on the non-autoregressive (NAR) model, which is a widely studied fast decoding method. We perform GEC in Japanese with traditional autoregressive and recent NAR models and analyze their accuracy and speed.

1 Introduction

Grammatical error correction (GEC) is a writing support method for language learners. In recent years, neural GEC has been actively researched owing to its ability to produce fluent text. For example, in Kiyono et al. (2019), state-of-the-art correction accuracy was achieved by using a Transformer (Vaswani et al., 2017), which is a powerful neural machine translation (NMT) model. Because the neural model can see the entire sequence, it can correct errors with long-range dependencies; these errors cannot be corrected by a statistical method that uses n-grams.

However, considering the application of GEC in a writing support system, we must consider how to handle incomplete sentences. It is easy to present the GEC result when the user finishes writing a sentence. However, in case of an incomplete sentence, the user will not know how to fix the sentence while writing it. If the system can perform GEC correctly for incomplete sentences, the results can be presented to the user. In most previous studies, complete sentences have been evaluated, and the performance of GEC for incomplete sentences has not been researched.

In addition, there is a problem that inference speed is slow in a conventional autoregressive (AR) decoder of a sequence-to-sequence model. Considering the application of GEC in a writing support system, a slower inference speed would restrict its utility or lower the usability of the model. In Gu et al. (2018), a non-autoregressive (NAR) decoder that speeds up inference time by outputting all tokens simultaneously was proposed. Following the success of NAR models, in Gu et al. (2019), Levenshtein Transformer, an NAR NMT model that iteratively deletes and inserts inputs, was proposed. Its usefulness was verified in machine translation and document summarization tasks.

Moreover, fast GEC methods with sequence tagging using an NAR model have been proposed. In Awasthi et al. (2019), GEC was regarded as a local sequence conversion task, and high-speed GEC was achieved by using an NAR model that iteratively adapted editing tags in parallel. In Omelianchuk et al. (2020), NAR GEC was performed by repetitive tagging of editing operations on each token of an input sentence, and higher correction accuracy and faster correction speed than in previous studies were achieved. However, these methods exhibited good performance by narrowing down the target language to English and preparing the editing operations as tags using language knowledge in advance.

In this study, we focus on the NAR model as a method for high-speed GEC. We perform GEC in Japanese using the NAR model that does not need to prepare editing operations in advance. We analyze the proposed method considering its application to writing support systems. In particular, we analyze the relationship between the correction

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 1–10
Suzhou, China, December 4, 2020.

accuracy and the inference speed, focusing on incomplete sentences, and evaluate the impact of hyperparameters on NAR models. The contributions of this study can be summarized as follows.

- We evaluate the performance of NAR and AR models for incomplete sentences in terms of accuracy and speed, aiming for the construction of a writing support system.
- We show that the Levenshtein Transformer that performs one-time iterative refinements can achieve fast and stable GEC by reducing the worst inference time by 6.0 seconds and the average inference time by 0.3 seconds compared with the method based on convolutional neural networks (CNNs).
- Using the NAR model for Japanese GEC, we find that it is better to present the GEC result when the number of input words is six or more because the accuracy is significantly reduced when the number is less than five.

2 Related Work

2.1 AR NMT

AR NMT is a standard decoding method in the encoder–decoder model (Kalchbrenner and Blunsom, 2013) for sequence-to-sequence learning. This method uses a recurrent language model (Mikolov et al., 2010) during inference.

Given an original sentence, $X = \{x_1, \ldots, x_{T'}\}$, and an objective sentence, $Y = \{y_1, \ldots, y_T\}$, an AR NMT model calculates the target sentence as

$$p(Y|X;\theta) = \prod_{t=1}^{T+1} p(y_t|y_{0:t-1}, x_{1:T'};\theta), \quad (1)$$

where y_0 and y_{T+1} are special tokens representing the beginning and end of the sentence, respectively, and θ is the model's parameter.

2.2 NAR NMT

NAR NMT (Gu et al., 2018) is a decoding method that generates each token independently and simultaneously. This method is attracting attention as a method to increase the speed of decoding.

In Gu et al. (2018), the concept of fertility, which predicts how many words on the target side correspond to each word in the source side, was introduced. The decoding is performed as follows:

$$p(Y|X;\theta) = \sum_{f_1,\ldots,f_{T'}\in\mathcal{F}} \left(\prod_{t'=1}^{T'} p_F(f_{t'}|x_{1:T'};\theta) \cdot \prod_{t=1}^{T} p(y_t|x_1\{f_1\},\ldots,x_{T'}\{f_{T'}\};\theta) \right), \quad (2)$$

where $\mathcal{F}$ is the set of all fertility sequences that sum into the length of Y, and $x\{f\}$ represents token x repeated f times. As described earlier, it is necessary to predict the target sentence length in the NAR decoding method.

Furthermore, NAR NMT involves a problem named the multimodality problem (Gu et al., 2018). This problem causes errors (such as token repetitions and a lack of tokens) and significantly deteriorates accuracy compared with an AR decoder. To solve this problem, in recent studies, iteratively refining the output (Lee et al., 2018; Gu et al., 2019) and partially autoregressively outputting the sentence divided into segments (Ran et al., 2020) have been proposed. Knowledge distillation (KD) (Kim and Rush, 2016) is also used to address this problem (Zhou et al., 2020). The output of the AR model is known to mitigate multimodality problems because diversity is suppressed such that the model can be easily learned (Ren et al., 2020).

2.3 Levenshtein Transformer

Levenshtein Transformer (Gu et al., 2019) is one of the most recent NAR NMT models[1] that introduces a workaround for the aforementioned multimodality problems. In Gu et al. (2019), the usefulness of the Levenshtein Transformer in machine translation and summarization tasks was verified; however, its usefulness in GEC has not been verified yet.

This model has a Transformer (Vaswani et al., 2017) block (T-block) as a primary component, and the original text is given to each T-block. First, the states coming from the lth T-block are as fol-

[1] In the original paper, it is called a "partially autoregressive model"; however, in this paper, we call it an NAR model because it is a model that outputs all tokens simultaneously when decoding.

lows:

$$\boldsymbol{h}_0^{(l+1)}, \boldsymbol{h}_1^{(l+1)}, \ldots, \boldsymbol{h}_n^{(l+1)} = \begin{cases} E_{y_0} + P_0, E_{y_1} + P_1, \ldots, E_{y_n} + P_n, & l = 0 \\ \text{T-block}_l\left(\boldsymbol{h}_0^{(l)}, \boldsymbol{h}_1^{(l)}, \ldots, \boldsymbol{h}_n^{(l)}\right), & l > 0 \end{cases} \quad (3)$$

where E and P are token and position embeddings, respectively; y_0 and y_n are boundary tokens representing the start and end, respectively. Next, we use these decoder outputs, $(\boldsymbol{h}_0, \boldsymbol{h}_1, \ldots, \boldsymbol{h}_n)$, to classify deletions, placeholders, and tokens. The deletion classifier uses $\text{softmax}\left(\boldsymbol{h}_i \cdot A^\top\right), (i = 1, \ldots n-1)$ to perform binary classification of "deleted" or "kept" for tokens other than boundary tokens. Next, it deletes corresponding tokens. The placeholder classifier uses $\text{softmax}\left(\text{concat}\left(\boldsymbol{h}_i, \boldsymbol{h}_{i+1}\right) \cdot B^\top\right), (i = 0, \ldots n-1)$ to classify how many placeholders to insert from 0 to $K_{\max}$ at every consecutive position pair. Subsequently, it inserts the corresponding number of special tokens <PLH>, where $K_{\max}$ is the maximum number of tokens that can be inserted at one time in one place, and we set it to 255. The token classifier uses $\text{softmax}\left(\boldsymbol{h}_i \cdot C^\top\right), (\forall y_i = \texttt{<PLH>})$ to classify and replace all special tokens <PLH> into words that are elements of vocabulary $\mathcal{V}$. Here, A, B, and C are matrices for linearly transforming the number of dimensions of a state or a combination of two states into the number of classes.

2.4 GEC

GEC is a task to correct errors, such as punctuation, grammar, and word selection errors. Various methods have been studied for this task. In recent years, owing to the development of NMT, GEC is often interpreted as a machine translation task. Almost all studies using the BEA Shared Task-2019 datasets (Bryant et al., 2019) used Transformer-based models (Omelianchuk et al., 2020; Kiyono et al., 2019; Kaneko et al., 2020; Grundkiewicz et al., 2019; Choe et al., 2019; Li et al., 2019). For example, in Li et al. (2019), a system that combined a CNN-based model with a Transformer-based model was used, and the method in Chollampatt and Ng (2018) was adopted as the CNN architecture.

The following are previous studies on high-speed GEC using an NAR model. In Awasthi et al. (2019), GEC was regarded as a local sequence conversion task, and it was rapidly solved by using a parallel iterative editing model. In Omelianchuk et al. (2020), the same task was solved by iterative sequence tagging. However, both methods applied linguistic knowledge prepared in advance (such as suffix conversion rules and verb conjugation dictionaries). Thus, it is not easy to apply them to another language. In this study, we propose a method that uses only a training corpus.

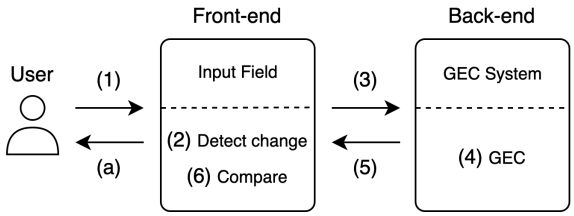


Figure 1: Schematic diagram of a writing support system.

3 Proposed Method

This study aims to analyze the effectiveness of an NAR model for Japanese GEC in terms of accuracy and speed, assuming that NAR is used as a back-end of a writing support system.

3.1 Writing Support System

In this section, we explain the workflow of a writing support system. Figure 1 shows a schematic diagram of the system, which consists of a front-end with a text input field and a back-end with a GEC system, and it works as follows. (1) The user inputs or deletes the text in the input field[2]. (2) The front-end detects the change; (3) it sends the changed sentence to the back-end. (4) The back-end performs GEC; (5) it sends the correction to the front-end. (6) The front-end checks for changes; (a) it suggests changes to the user if there are changes; (b) otherwise, it does nothing.

3.2 Challenges for the System

In this section, we consider the system input ((1) and (2)) and response time ((2) to (6a)).

System Input We consider two problems with input from users of the system.

The first is how to process a sentence in the middle of input. When the user finishes writing a sentence (in other words, when the user enters a line),

[2]For simplicity, we assume that the user enters one sentence per line. In other words, line breaks divide the sentences.

the GEC result of the sentence should be presented to the user. However, it is unclear how to deal with an incomplete sentence. This is because the back-end system may not perform accurate GEC owing to its incompleteness or shortness. Thus, we propose the following hypothesis: if the incomplete sentence is short, the correction accuracy deteriorates; however, if it is long, the correction accuracy approaches that of the complete sentence. We verify this hypothesis in Subsection 4.2.

The second is the problem of the Japanese input method. In Japanese, unlike English, user inputs are processed through a kana–kanji conversion system[3]. In other words, when the front-end is receiving the text through the kana–kanji conversion, it is not evident in what unit (character, word, phrase, or whole sentence) the errors can be appropriately detected[4]. In this study, we assume that users are intermediate Japanese learners and treat the input string as words.

Response Time The processing speed from (2) to (6a) in the system flow dominates the response time of the system, which affects the user experience. It is known that not only is responsiveness required, but also users prefer a system with constant response speed over a system with variable response speed (Shneiderman, 1979). We analyze the processing time of GEC in Subsection 4.3.

4 Experiment

4.1 Experimental Settings

Dataset We use data from the Lang-8 learner corpus (Mizumoto et al., 2011). We use the TMU Evaluation Corpus for Japanese Learners (Koyama et al., 2020) for the validation and test sets[5]. All data, including the training set, are pre-processed as in Koyama et al. (2020). Table 1 presents the number of sentences in the data. We use the same training set in our experiments with both complete and incomplete sentences.

To evaluate the performance of GEC for incomplete sentences, we segment the test data to the word level and then create incomplete sentences by increasing the number of words from the beginning. For example, 10 sentences are created from a 10-word sentence. Next, based on the word alignment between the source and target sentences, we create parallel sentences for incomplete sentences. Consequently, 9,710 sentence pairs are created. We use these data to evaluate the performance of GEC for incomplete sentences.

	# of sentences	# of corrections
Train	1,093,633	1
Validation	806	2
Test	663	3

Table 1: Dataset statistics. The number of corrections denotes the number of reference sentences for one learner's sentence.

Tokenization We tokenize data in all models as follows. First, we segment data into morpheme units using MeCab[6] (Ver. 0.996) using the UniDic[7] (Ver. 2.2.0) as a dictionary. Next, we divide the morpheme units into subword units by applying the byte pair encoding (Sennrich et al., 2016) model for dealing with rare words. We apply character normalization (compatibility decomposition, followed by canonical composition) and share vocabulary between source and target sides.[8] The vocabulary size was set to 30,000 words. We use sentencepiece[9] for implementation.

NAR Model In this study, we apply the Levenshtein Transformer (Gu et al., 2019), which is a Transformer-based NAR neural model, to GEC. We update the model 300,000 times with a batch size of 64,000 tokens and select the model with the highest GLEU score (Napoles et al., 2016) for the validation set. Other hyperparameters are the same as in Gu et al. (2019). We use publicly available PyTorch-based code[10] for implementation. In this paper, this model is called the LevT model.

The maximum number of iterative refinements was set to nine in a previous study (Gu et al.,

[3]The kana–kanji conversion system translates the input hiragana (the Japanese cursive syllabary) into kanji (Chinese characters) when necessary.

[4]The appropriate unit may change depending on the user's language learning level and Japanese input ability level.

[5]These data are less noisy than the corrected sentences included initially in the Lang-8 learner corpus and have multiple references to all sentences, which is considered useful for evaluation.

[6]https://taku910.github.io/mecab/

[7]https://unidic.ninjal.ac.jp/

[8]As a preliminary experiment, the source side was set to the character unit, and the target side was set to the subword unit; however, the GLEU score (Napoles et al., 2016) was slightly decreased; therefore, we decided to tokenize both sides in the subword units.

[9]https://github.com/google/sentencepiece

[10]https://github.com/pytorch/fairseq/tree/master/examples/nonautoregressive_translation

2019). However, it is unclear whether it is the correct value for GEC because GEC is a local sequence conversion task in which almost all the source words remain in the target side. Therefore, we also evaluate the performance when the maximum iterative refinement number is changed.

Training data are obtained by replacing the corrected sentences with the output of an AR model. We use it to train a KD model (Zhou et al., 2020) of LevT. The hyperparameters are the same as for the LevT model, and the model described in the next paragraph is used for the AR model. In this paper, this model is called the LevT+KD model.

AR Model Because we focus on speeding up GEC, we adopt the CNN-based model (Chollampatt and Ng, 2018), which is faster than the Transformer-based model as the AR baseline. Unlike Chollampatt and Ng (2018), the output is not reranked to match the conditions with the LevT model. Other hyperparameters are the same as in Chollampatt and Ng (2018). We use publicly available PyTorch-based code[11] for implementation. In this paper, this model is called the CNN model.

Correction Evaluation We use the GLEU score (Napoles et al., 2016) and the $F_{0.5}$ score, which weighs the precision as twice the recall, as evaluation metrics for the correction accuracy of GEC. We map words automatically using the ERRANT[12] to calculate $F_{0.5}$. However, because the ERRANT is designed for English, we cannot use it directly; instead, we specify the Levenshtein distance in the distance function that calculates the edit distance without using linguistic information. Furthermore, we measure the $F_{0.5}$ score in terms of the word-wise agreement.

Inference Speed Evaluation We measure the inference speed with the following settings. We use the Intel® Xeon® processor E5-2660 without GPUs. To measure the performance in a realistic setting, we set the batch size to one sentence. We measure time using the built-in `time` module in Python. Specifically, the inference speed of one sentence is calculated as the change in the system clock from the input of the sentence before tokenization to the output of the GEC result.

Model	GLEU	Prec.	Rec.	$F_{0.5}$
CNN	73.3	0.159	**0.225**	0.169
LevT	72.1	0.102	0.185	0.112
LevT+KD	**75.2**	**0.213**	0.217	**0.214**

Table 2: Correction accuracy of each model for complete sentences from the test set. "Prec." and "Rec." represent precision and recall, respectively.

Model	M (242)	R (441)	U (124)
CNN	**33**	73	**32**
LevT	22	54	5
LevT+KD	**33**	**79**	20

Table 3: Number of errors, according to category, that each model modified correctly on the test set. "M," "R," and "U" represent missing, replacement, and unnecessary errors, respectively. Each number in parentheses represents the maximum error frequency[13].

4.2 Correction Accuracy

To confirm GEC's effectiveness on complete sentences, we evaluate each model using the test set.

Table 2 lists the results. Both GLEU and $F_{0.5}$ scores of LevT without KD are worse than those of CNN, but LevT+KD's score exceeds CNN's score. In terms of the recall and precision of LevT and LevT+KD, both are improved by KD, and the precision is significantly increased. Therefore, KD dramatically improves the precision in GEC.

For a more detailed analysis of the effect of KD, the number of categorical errors that the model correctly changed is presented in Table 3. Comparing LevT and LevT+KD, it can be seen that the number of corrections for all types of errors has increased owing to KD. In particular, the correction accuracy for "unnecessary" errors has increased. We believe that this is because LevT+KD can inherit the correction accuracy for the "unnecessary" errors of the CNN by KD.

LevT+KD has a correction accuracy comparable to that of the CNN. As KD's effectiveness in the NAR model for GEC is confirmed, we focus only on LevT+KD for the NAR model in the subsequent experiments.

[11] https://github.com/nusnlp/mlconvgec2018

[12] https://github.com/chrisjbryant/errant

[13] The maximum number of corrections made by each of the three annotators is shown. The smallest ones are 210, 428, and 103.

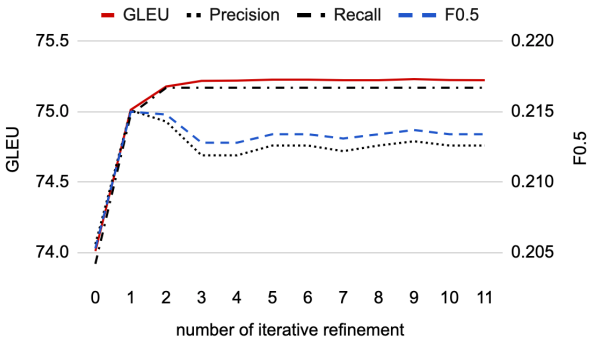


Figure 2: Accuracy of LevT+KD model for the maximum number of iterative refinements. The solid red, dotted black, dash-dotted, and broken blue-black lines represent the GLEU score, precision, recall, and $F_{0.5}$ score, respectively.

Model	GLEU	Prec.	Rec.	$F_{0.5}$
CNN	74.6	0.100	**0.199**	0.111
LevT+KD	**76.9**	**0.125**	0.184	**0.133**

Table 4: Correction accuracy of each model for incomplete sentences.

Number of Iterative Refinements Unlike in the machine translation task (which was mainly addressed in the previous studies on NAR models), it seems that the necessary number of iterative refinements is reduced in the GEC task because the input and the output are close. Therefore, we evaluate the change in performance because of the number of iterative refinements in the LevT+KD model.

Figure 2 shows the result of the GLEU and $F_{0.5}$ scores for the best epoch selected in the validation set. We can see that after the first iteration, the GLEU score does not change significantly with the maximum number of iterative refinements, and it is almost optimal when the number is three. Furthermore, similar to the GLEU score, we can see that the change in the $F_{0.5}$ score after the first iteration is small. Moreover, when the number of iterations is more than one, the score degrades with a decrease in precision. When the maximum number of iterations is one, the score becomes the maximum. Therefore, there is little need to increase the maximum number of iterative refinements of LevT+KD in GEC. One to three iterations are sufficient.

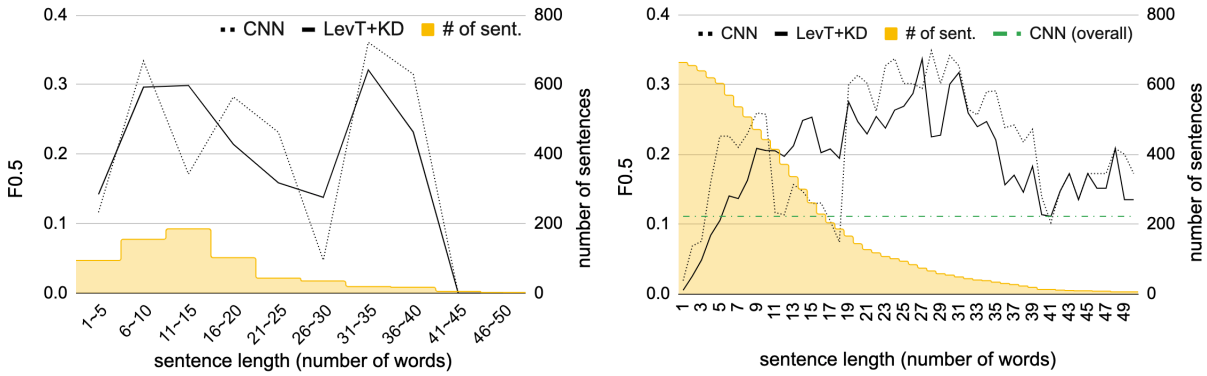
Accuracy for Incomplete Sentences Table 4 shows each model's overall correction accuracy for incomplete sentences. Compared with Table 2, it can be seen that the overall tendency is the same: LevT+KD has a high GLEU score and a high precision, whereas CNN has a high recall. Furthermore, in both models, the GLEU score improves slightly, and the $F_{0.5}$ score deteriorates for incomplete sentences. Overall, the GEC model trained only on complete sentences is useful to some extent, even for incomplete sentences.

Figure 3 shows each model's correction accuracy per sentence length. Comparing the incomplete sentences (b) with the complete sentences (a), we can see that the accuracy is considerably reduced when the sentence length is extremely short in both models. When presenting the GEC result for incomplete sentences, it is considered appropriate not to show it when the input sentence length is short. In addition, the correction accuracy for the complete sentences fluctuates substantially in the range of 31–50 words in both models. This might be attributed to the lack of test sentences.

4.3 Inference Speed

Figure 4 shows the inference speed of test data containing 9,710 incomplete sentences for each model. The average inference times are 0.49, 0.24, and 0.19 seconds for CNN, LevT+KD with the maximum number of iterations set as nine, and LevT+KD with the maximum iterations set as one, respectively. According to our results, the variance of the inference time of LevT+KD is significantly suppressed compared with that of CNN, and the average time is also significantly lower than that of CNN. The variance and average can be further suppressed by reducing the maximum number of iterations. Excluding the outliers, CNN also fits in approximately one second. However, the correspondence between the inference speeds of each model does not change, and LevT+KD is faster than CNN. Here, most sentences with outliers are long sentences created from sentences whose original length is 100 words or more, and we believe that the lengthy sentences are the leading cause of the increase in inference time.

Figure 5 depicts the inference speed for each sentence length of each model. Focusing on the linear approximation, we find that CNN is faster than each LevT+KD when the sentence length is extremely short (one to four words). We assume that this is because the Levenshtein Transformer executes three types of operations: delete, insert a



(a) For complete sentences.

(b) For incomplete sentences.[14]

Figure 3: Correction accuracy per sentence length breakdown for complete and incomplete sentences. The solid, dotted, and straight dash-dotted green lines represent the $F_{0.5}$ scores of LevT+KD and CNN, and CNN's overall $F_{0.5}$ score, respectively. The step-form graph represents the number of sentences.

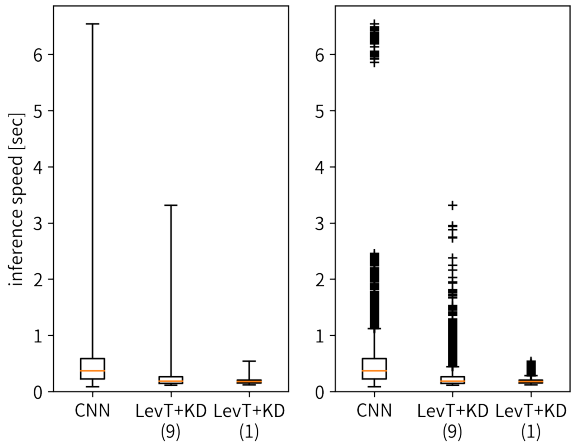


Figure 4: Inference speed of each model. The number in parentheses in the model name represents the maximum number of iterations. The graph on the left does not consider outliers, and the graph on the right shows outliers as "+" in the range where the whiskers length exceeds 1.5 times the interquartile range.

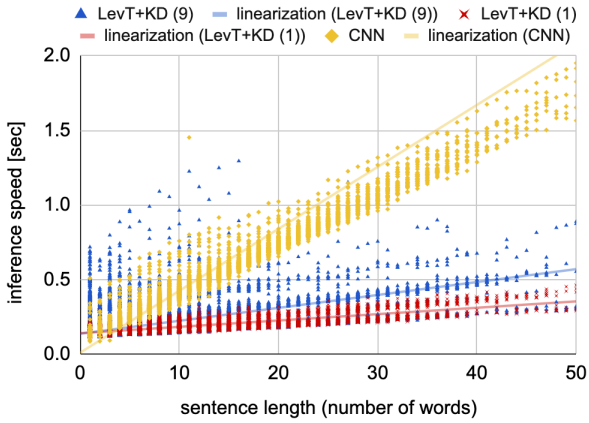


Figure 5: Inference speed of each model per sentence length breakdown. Each straight line represents a linear approximation, and the number in parentheses in the model name represents the maximum number of iterations.

placeholder, and replace it with a token in one iterative refinement, thereby having more overhead than the CNN model does. However, the results show that each LevT+KD model is faster than CNN when the sentence length is five words or more.

Here, we analyze the effect of sentence length on incomplete sentences in terms of both correction accuracy and speed. As shown in Figure 3b, when CNN's overall $F_{0.5}$ score for the incomplete sentence is used as the minimum criterion, the LevT+KD model's scores with five or fewer words are below the standard. Furthermore, Figure 5 shows that LevT+KD is consistently faster than CNN after six words or more. In other words, by using the LevT+KD model and performing GEC when six or more words are input, it is possible to present the correction results at high speed while maintaining a certain degree of correction accuracy[15].

4.4 Case Study

We show examples of system output in Table 5. In (1), a sentence in which "ですか? *desuka?*"

[14]Note that the performance is stable because more sentences are used to evaluate the GEC model than complete sentences.

[15]Sentences of five or fewer words account for approximately 32.7% of the incomplete sentences used in this experiment. Furthermore, in reality, considering that long sentences are corrected many times, it is believed that the rate of short sentences of fewer than five words is even lower.

(1)	Learner's sentence	これはほんとに大切だか?
	CNN	これはほんとに大切**なの**?
	LevT+KD	これはほんとに大切**です**か?
	Corrected sentence	これはほんとに大切だろうか? "Is this really important?"
(2)	Learner's sentence	かわいいくて、安い、素敵な生地です。
	CNN	**かわいいて**、安い、素敵な生地です。
	LevT+KD	**かわいて**、安い、素敵な生地です。
	Corrected sentence	かわいくて、安い、素敵な生地です。 "It's a cute, cheap and lovely fabric."

Table 5: Output examples for each model. Grammatical errors are underlined. Boldface represents where the model has changed the text. Double quotes represent the meaning of the sentence.

Input: learner's sentence			き の よる は たく やき パーチイ ー い ます 。										
insert[17](0)	きのう		よる	は	たくさん		パーティー	パーティー		し	ます	。	
delete (1)	きのう		よる	は	たくさん		パーティー	~~パーティー~~		し	ます	。	
insert (1)	きのう		よる	は	たくさん	やき	パーティー		を	し	ます	。	
delete (2)	きのう		よる	は	たくさん	やき	パーティー		を	し	ます	。	
insert (2)	きのう	の	よる	は	たくさん	やき	パーティー		を	し	ます	。	
	kinou	*no*	*yoru*	*wa*	*takusan*	*yaki*	*paatii*	*paatii*	*wo*	*shi*	*masu*		
System output sentence	き の う の よ る は たくさん やき パーティー を し ます 。 "I have a lot of bake party last night."												
Corrected sentence	き の う の よ る は た こ やき パーティー に い ま し た 。 "I was at a Takoyaki party last night."												

Table 6: Example of iterative refinements. The number of iterations is written in parentheses. Grammatical errors are underlined. Boldface represents the inserted word, and strikethrough represents the deleted word. Italics represent the Japanese pronunciations of each word, and double quotes represent the meaning of the sentence.

is mistaken for "だか? *daka?*"[16] is input. The correction differs depending on the model, but the outputs of both models are grammatically correct. In (2), a sentence in which "かわいく *kawaiku*," which is a conjunctive form of "かわいい *kawaii*" (cute), is mistaken for "かわいいく *kawaiiku*" is input. Both models changed the error part, but both outputs are grammatically incorrect. We believe that the reason for this is that there are few similar error examples in the training set. In the training set, there are 172 errors of "だか? *daka?*," whereas only two errors of "かわいいくて *kawaiikute*." We assume that the performance of the sequence-to-sequence GEC method is limited by the number of similar errors in the training set.

Table 6 presents an example of iterative refinements in LevT+KD. In the first insertion phase, a missing token error and repeated token errors have occurred. The repeated token, "パーティー *paatii*" (party), is deleted in the next deletion, and in the next insertion phase, the missing tokens, "やき *yaki*" (bake) and "を *wo*" (accusative case marker), are inserted to the left and right of "パーティー *paatii*," recovering from the multimodality problem. However, erroneous parts remain: "たくやき *takuyaki*," which is a misspelling of "たこやき *takoyaki*" (octopus dumplings), is mistakenly corrected as "たくさん やき *takusan yaki*." Moreover, "ます *masu*" (politeness marker), which should be corrected to the past form corresponding to "きのう *kinou*" (yesterday), is not corrected.

5 Conclusion

In this study, we investigated the applicability of the NAR model, which has a constant inference speed, to Japanese GEC toward constructing a writing support system. The experiments showed that the NAR model can obtain a correction accuracy that is equal to or better than that of the AR multilayer convolutional neural model. Furthermore, we demonstrated that the GEC model trained on complete sentences can also be applied to incomplete sentences. However, we found that when the number of input words is small, the correction accuracy is significantly lower than that of the complete sentence. Therefore, the system should defer presenting correction results for short

[16] The Japanese question marker particle, "か *ka*," cannot be added at the end of a sentence in the plain-style sentence.

[17] `insert` shows the result of both inserting the placeholder and replacing it with the actual token. In addition, because it starts with an empty string, there is no `delete` in the first iteration.

sentences. We also showed that the worst inference time could be reduced by approximately 6.0 seconds, and the average inference time could be reduced by approximately 0.3 seconds in the NAR model that performs one-time iterative refinement compared with the AR model.

Future work includes an extrinsic evaluation of the GEC system integrated into a writing support system. Moreover, we plan to investigate a large-scale pretrained model to improve GEC's performance.

Acknowledgments

We want to thank Yangyang Xi for consenting to use text from Lang-8.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. 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 4260–4270. Association for Computational Linguistics.

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. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence*, pages 5755–5762. Association for the Advancement of Artificial Intelligence.

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. Association for Computational Linguistics.

Jiatao Gu, James Bradbury, Caiming Xiong, Victor O.K. Li, and Richard Socher. 2018. Non-autoregressive neural machine translation. In *International Conference on Learning Representations*.

Jiatao Gu, Changhan Wang, and Junbo Zhao. 2019. Levenshtein transformer. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *Advances in Neural Information Processing Systems 32*, pages 11181–11191. Curran Associates, Inc.

Nal Kalchbrenner and Phil Blunsom. 2013. Recurrent continuous translation models. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1700–1709. Association for Computational Linguistics.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4248–4254. Association for Computational Linguistics.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing*, pages 1317–1327. Association for Computational Linguistics.

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. Association for Computational Linguistics.

Aomi Koyama, Tomoshige Kiyuna, Kenji Kobayashi, Mio Arai, and Mamoru Komachi. 2020. Construction of an evaluation corpus for grammatical error correction for learners of Japanese as a second language. In *Proceedings of The 12th Language Resources and Evaluation Conference*, pages 204–211. European Language Resources Association.

Jason Lee, Elman Mansimov, and Kyunghyun Cho. 2018. Deterministic non-autoregressive neural sequence modeling by iterative refinement. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1173–1182. Association for Computational Linguistics.

Ruobing Li, Chuan Wang, Yefei Zha, Yonghong Yu, Shiman Guo, Qiang Wang, Yang Liu, and Hui Lin. 2019. The LAIX systems in the BEA-2019 GEC shared task. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 159–167. Association for Computational Linguistics.

Tomas Mikolov, Martin Karafiát, Lukás Burget, Jan Cernocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association*, pages 1045–1048. International Symposium on Computer Architecture.

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. Asian Federation of Natural Language Processing.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. GLEU without tuning. *eprint arXiv:1605.02592 [cs.CL]*.

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. Association for Computational Linguistics.

Qiu Ran, Yankai Lin, Peng Li, and Jie Zhou. 2020. Learning to recover from multi-modality errors for non-autoregressive neural machine translation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 3059–3069. Association for Computational Linguistics.

Yi Ren, Jinglin Liu, Xu Tan, Zhou Zhao, Sheng Zhao, and Tie-Yan Liu. 2020. A study of non-autoregressive model for sequence generation. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 149–159. Association for Computational Linguistics.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725. Association for Computational Linguistics.

Ben Shneiderman. 1979. Human factors experiments in designing interactive systems. *Computer*, 12(12):9–19.

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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Chunting Zhou, Jiatao Gu, and Graham Neubig. 2020. Understanding knowledge distillation in non-autoregressive machine translation. In *International Conference on Learning Representations*.

Arabisc: Context-Sensitive Neural Spelling Checker

Yasmin Moslem, Rejwanul Haque and Andy Way
ADAPT Centre
School of Computing
Dublin City University
Dublin, Ireland
firstname.lastname@adaptcentre.ie

Abstract

Traditional statistical approaches to spelling correction usually consist of two consecutive processes – error detection and correction – and they are generally computationally intensive. Current state-of-the-art neural spelling correction models usually attempt to correct spelling errors directly over an entire sentence, which, as a consequence, lacks control of the process, e.g. they are prone to overcorrection. In recent years, recurrent neural networks (RNNs), in particular long short-term memory (LSTM) hidden units, have proven increasingly popular and powerful models for many natural language processing (NLP) problems. Accordingly, we made use of a bidirectional LSTM *language model (LM)* for our context-sensitive spelling detection and correction model which is shown to have much control over the correction process. While the use of LMs for spelling checking and correction is not new to this line of NLP research, our proposed approach makes better use of the rich neighbouring context, not only from before the word to be corrected, but also after it, via a *dual-input* deep LSTM network. Although in theory our proposed approach can be applied to any language, we carried out our experiments on Arabic, which we believe adds additional value given the fact that there are limited linguistic resources readily available in Arabic in comparison to many languages. Our experimental results demonstrate that the proposed methods are effective in both improving the quality of correction suggestions and minimising overcorrection.

1 Introduction

Misspelling detection or/and correction modules are seen as critical components of many real-world NLP applications. This has also been regarded as an important research area of NLP for years. The spelling errors are broadly classified into two categories: non-word errors (NWE), and real-word errors (RWE). If the misspelled string is a valid word of a language, it is called an RWE, otherwise it is an NWE (Choudhury et al., 2007). In this context, Peterson (1986) found that the RWE rate ranges from 2% for a small lexicon to 10% for a 50,000-word lexicon and almost 16% for a 350,000-word lexicon. In this work, we investigate both error types (i.e. RWE and NWE) with our context-aware spelling error detection and correction models. We demonstrate that our approach is capable of detecting and correcting both NWEs and RWEs in a text. As an illustration, we present two sentences that contain misspelled words below, with a justification of why context-sensitive error detection and correction could be an ideal solution for this problem.

English:

- **Wrong:** Students met their *Principle* Supervisor at the University.
- **Correct:** Students met their *Principal* Supervisor at the University.

Arabic:

- **Wrong:** هذه هي الطريق المثلي التي يجب أن نتخذها
- **Correct:** هذه هي الطريق المثلى التي يجب أن نتخذها

In the first English sentence, we can see that the word *Principle* is a correct word that we can find in a dictionary; however, its use in this context is incorrect and the right word in this context is to be *Principal*. Hence, we can call this an RWE, and we can clearly see that this requires help from the neighbouring lexical contexts for error detection and correction. Similarly, in the Arabic example, we can see that the adjective المثلي (*almthly*) was incorrectly used instead of المثلى (*almthla*) to describe الطريق (*altariq*). Like the error in the English example, this error requires the same treatment.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 11–19
Suzhou, China, December 4, 2020.

Traditional rule-based and statistical approaches to spelling correction rely on error detection first before offering correction suggestions. This minimises the chances of making unrequired corrections at least for common words. However, creating a good spelling checker using such traditional approaches involves building a large lexical database and thousands of human-generated rules for NWEs, or large phrase tables for RWEs (Verberne, 2002). This, in effect, requires a lot of linguistic resources and tools as well as massive computing resources.

Many neural approaches (Weiss, 2016) to spelling checking normally correct errors directly over an entire input sentence. Presenting an entire sentence to the network or decoder for correction involves the risk of modifying words that are correct in the context and should not be changed. For instance, the experiments carried out by Weiss (2016) demonstrate how neural spelling checking models can make overcorrection mistakes with examples. They categorise such errors as follows (Q: input; A: ground truth; S: system output):

1- Correcting words that are not really misspellings:

- Q. In addition to personal-injury and
- A. In addition to personal-injury and
- S. In addition to personal injury and

2- Changing the original meaning:

- Q. had learned of Ca secret plan y Iran
- A. had learned of a secret plan by Iran
- S. had learned of a secret plan I ran

3- Even introducing new misspellings:

- Q. post-Thanksgiving performances, but
- A. post-Thanksgiving performances, but
- S. post-thanks gving performances, but

As can be seen from these examples, the neural model corrects some words that should not be corrected. We conjecture that this happened because the model tries to make correction directly on the entire sentence while bypassing the error detection process. In this context, Hertel (2019) found that neural *many-to-many* encoder-decoder models for spelling correction perform worse than neural *many-to-one* LM-based approaches. What if we rather ask the neural network to first "detect" the error and then "correct" it, with the help of language modelling while still taking the context into consideration? This is the research question we explore in this paper.

In this work, we propose a context-sensitive neural model, *Arabisc*,[1] which adds more control to the spelling correction process using language modelling, i.e. a many-to-one LSTM network, and it consists of two processes: (i) identifying spelling errors, and (ii) offering correction suggestions. The idea is that we have to only correct the mistakes not the whole sentence. In other words, we combine the best of two worlds (statistical[2] and neural) i.e. we detect potential spelling mistakes and then offer diverse correction suggestions for the user to choose from in one go.[3] Although we tested our method on standard Arabic (*Fosha*), it can theoretically be applied to any other language.

The rest of this paper is organised as follows. Section 2 elaborates on our methodology including the architecture of our proposed model. Section 3 describes the experimental results and findings with some discussions, while Section 4 concludes and suggests some avenues for future work.

2 Methodology

The backbone of our approach involves building and using language modelling, i.e. a *many-to-one* LM for text generation. The task is to check a given input sentence *word by word*, predict the next word, and find out whether the current word cw of the input sentence is in the list of high-scoring candidates B generated by the LM given the context of cw (previous and following words of cw). If cw is not in the list, correction suggestions are offered based on the edit distance (Levenshtein, 1966) between cw and the candidates in B. In our work, we compare two different models, namely: (i) a single-input model that uses only the preceding words of cw as context, and (ii) a dual-input model that uses the preceding and following words of cw as context. We describe our models in detail in the following section.

2.1 Experimental Setups

2.1.1 Training Data

In order to build an LM to be used in the spelling correction task, it is important to make sure that

[1] *Arabisc* is a common misspelling of the word *Arabesque*, which refers to a form of artistic decoration. Surprisingly, *Arabisc* (or *Arabiśc*) is a real word from old English and it means Arabic or an Arab. Wikipedia: https://en.wiktionary.org/wiki/Arabisc

[2] Neural networks are statistical models. In this paper, we use "statistical" to refer to those models that do not have neural components.

[3] In our implementation, suggestions are generated as a JSON object, which can be used to display correction options to users, i.e. via a GUI.

sentences in our training set are linguistically correct and do not have many spelling mistakes. We selected the News Commentary Corpus v11[4] from OPUS (Tiedemann, 2012) as it is a reasonably clean corpus. We applied the standard filtering and pre-processing steps to the corpus. We are left with 213,036 Arabic sentences after cleaning and pre-processing. We also added a portion of the MultiUN[5] corpus from OPUS to the News Commentary corpus. Our final training data contains 554,622 Arabic sentences. The MultiUN corpus is of a better linguistic quality and the News Commentary corpus is more generic in nature. Therefore, we believe that adding the MultiUN corpus to the News Commentary corpus enriches our training data vocabulary. In order to pre-process the training sentences, we applied the following steps:

- Split those lines that consist of multiple segments based on newline, period followed by a space or a newline, Arabic question mark "؟", and exclamation mark;
- Remove duplicate segments;
- Remove Arabic diacritics, mainly *Tashkil* (marks used as phonetic guides);
- Remove punctuation marks and numbers. Some spelling checkers would keep punctuation marks and even correct them; but for the purpose of our experiments, we chose to remove them;
- Remove Latin characters;
- Append a start token <s> at the beginning;
- In order to avoid repetitions after applying the next step, truncate the sentences up to the maximum sequence length; and
- For our single-input encoder (cf. Section 2.2.1), generate *n*-gram sequences, using all preceding tokens as the context except the current token (cw) which is used as the label. Tables 1 and 2 illustrate the *n*-gram generation process. As for our dual-input encoder (cf. Section 2.2.2), in addition to the preceding tokens, include the remaining tokens after the label (cw) as the context, in *reverse* order, as the second contextual input. The *n*-gram generation process of the latter setup is illustrated in Tables 3 and 4.

[4] http://opus.nlpl.eu/News-Commentary.php

[5] http://opus.nlpl.eu/MultiUN.php

Input Sentence	
<s> students met their principle supervisor at the university	
Initial Sequence	**Current Word**
<s>	students
<s> students	met
<s> students met	their
<s> students met their	principle
<s> students met their principle	supervisor
<s> students met their principle supervisor	at
<s> students met their principle supervisor at	the
<s> students met their principle supervisor at the	university

Table 1: Single-input *n*-gram splitting of an English sentence.

Input Sentence	
<s> هذه هي الطريق المثلي التي يجب أن تتخذها	
Initial Sequence	**Current Word**
<s>	هذه
<s> هذه	هي
<s> هذه هي	الطريق
<s> هذه هي الطريق	المثلي
<s> هذه هي الطريق المثلي	التي
<s> هذه هي الطريق المثلي التي	يجب
<s> هذه هي الطريق المثلي التي يجب	أن
<s> هذه هي الطريق المثلي التي يجب أن	تتخذها

Table 2: Single-input *n*-gram splitting of an Arabic sentence.

2.1.2 Evaluation Test Set

In order to evaluate *Arabisc*, our spelling correction model, we randomly extracted 20 Arabic unseen sentences from the UN corpus.[6] From now on, we refer to this set of sentences as the evaluation test set. We introduced two types of errors in our evaluation test set: (i) the first set contains RWEs based on the confusion lists provided by Al-Jefri and Mahmoud (2013), and (ii) the second set contains NWEs based on deletion, insertion, substitution and transposition of adjacent alphabets in a word, being the causes of most spelling errors (Damerau, 1964). In our experiments, we used a development set which has helped us explore potential issues in relation to Arabic spelling checking and correction and fine-tune hyper-parameters.

Each sentence of the test set was pre-processed the way we prepared the training corpus (cf. Section 2.1.1). We split each test set sentence into a list of initial *n*-gram sequences and use the last word as the current word (cw) that we want to compare with the high-scoring next-word candidates *B* generated by the LM. Tables 1 and 2 demonstrate the *n*-gram generation process for the single-input decoder. As for the multiple input decoder, we provide the model with two sets of input tokens, i.e. tokens before and after the current word (cw) to be checked, and the feature generation process is shown in Tables 3 and 4.

[6] http://opus.nlpl.eu/UN.php

2.2 Arabisc

2.2.1 Single-Input Encoder

Our *many-to-one* spelling correction model is an RNN (Rumelhart et al., 1986; Werbos, 1990) with LSTM units (Hochreiter and Schmidhuber, 1997). The total number of layers in the network is 4. We use an embedding layer with an input dimension 256 and then add two hidden layers, one bidirectional LSTM with 512 units followed by an LSTM with 128 units. The output layer is a *Dense* layer with the softmax activation function, and the number of units in this layer is equal to the vocabulary size. The model is trained with the *Adam* optimizer (Kingma and Ba, 2015), with the learning-rate set to 0.001. The sparse categorical cross-entropy is used as the loss function. As mentioned earlier, we limit the maximum sequence length to 15 tokens.[7] The vocabulary size is set to 100,000 of the most frequently occurring tokens in the corpus. The encoder takes an input in the form of *n*-gram sequences generated by the training example creation module described in Section 2.1.1. For building our network, we used Keras Sequential API of TensorFlow 2.[8] The model was trained on 2 GeForce RTX 2080 TI GPUs for 8 epochs. Early stopping was used on the validation accuracy. In this setup, we found that the training loss was 4.88 and training accuracy was 0.26.

2.2.2 Dual-Input Encoder

We start this section by revisiting the example sentence *"Students met their Principal Supervisor at the University,"* and the list of conditional contexts (i.e. *n*-grams) shown in Table 3. We can see from the table that the word *"Principal"* is affected by words before it (e.g. *"Students"*) and words after it (e.g. *"Supervisor"* and *"University"*). Therefore, using a dual-input encoder that takes both the preceding and following contexts into account can be more appropriate as far as the spelling error detection and correction are concerned. Note that our dual-input encoder is similar in terms of its architecture to the single-input encoder. The only difference is that the conditional context of the word (cw) to be predicted comprises two inputs: the tokens ($[w_1, w_2 ... w_{n-1}]$) that come before the current word cw, and the tokens ($[w_{n+1}, w_{n+2} ...]$) that come after the current word cw in *reverse* order. To exemplify, for the aforementioned sentence, we will have:

Left-Branch Input: <s> → students → met → their

Current Word: → principal ←

Right-Branch Input: supervisor ← at ← the ← university

As we can see above, both the preceding and following parts of the input sequence are used as the conditional context by the neural network for the prediction of the token in between. We apply the same step to all tokens to be predicted. We conducted experiments by both keeping and reversing the order of tokens in the right-branch input, and found that the model with reversing the tokens that follow the current word cw beforehand works best in terms of the validation and test set accuracy. Note that Section 2.1.1 describes the details of pre-processing the input data.

In this setup, we used Keras Functional API of TensorFlow 2, that allows multiple inputs. The identical four layers described in Section 2.2.1 are used for each of the two inputs. Finally, the two output layers are merged together using a *Concatenate* layer to generate the final (single) output using a *Dense* layer. Figure 1 illustrates the right and left branches of our dual-input neural network. Like the single-input encoder, the dual-input model was trained on 2 GeForce RTX 2080 TI GPUs for 11 epochs. Early stopping was used on the validation accuracy. In this setup, we found that the training loss was 3.15 and training accuracy was 0.46.

2.2.3 Bidirectional LSTM versus Dual-Input Encoder

In Section 2.2.1, we pointed out that the Single-Input Encoder uses a *Bidirectional LSTM* layer. If we express this in a different way, the bidirectional effect is applied only up to the word that is currently being generated, i.e. the "Left-Branch Input" in the aforementioned example. As for the Dual-Input Encoder described in Section 2.2.2, in addition to the "Left-Branch Input", it uses the "Right-Branch Input" which plays a pivotal role in observing the wider context and improving the quality of spelling corrections. This subtlety differentiates fundamental single-input language modelling from the encoder-decoder architecture, as the former takes only words before the current word to be generated while the latter deals with the sentence as a whole. As pointed out earlier, using *many-to-one* text generation with LMs for spelling correction tasks brings about better quality over using *many-to-many* encoder-decoder architectures (Hertel, 2019). Hence, we chose to use language modelling to have more control over the correction process, word by word, while we propose to use the Dual-Input Encoder to solve this limitation. While

[7] We restricted the length to 15 as processing longer sentences is found to be computationally expensive.

[8] https://github.com/tensorflow/tensorflow

Input Sentence		
<s> students met their principle supervisor at the university		
1st Input Sequence	**Current Word**	**2nd Input Sequence (in reverse order)**
<s>	students	university the at supervisor principle their met
<s> students	met	university the at supervisor principle their
<s> students met	their	university the at supervisor principle
<s> students met their	principle	university the at supervisor
<s> students met their principle	supervisor	university the at
<s> students met their principle supervisor	at	university the
<s> students met their principle supervisor at	the	university
<s> students met their principle supervisor at the	university	

Table 3: Dual-input *n*-gram splitting of an English sentence.

Input Sentence		
<s> هذه هي الطريق المثلي التي يجب أن نتخذها		
Initial Sequence	**Current Word**	**2nd Input Sequence (in reverse order)**
<s>	هذه	نتخذها أن يجب التي المثلي الطريق هي
<s> هذه	هي	نتخذها أن يجب التي المثلي الطريق
<s> هذه هي	الطريق	نتخذها أن يجب التي المثلي
<s> هذه هي الطريق	المثلي	نتخذها أن يجب التي
<s> هذه هي الطريق المثلي	التي	نتخذها أن يجب
<s> هذه هي الطريق المثلي التي	يجب	نتخذها أن
<s> هذه هي الطريق المثلي التي يجب	أن	نتخذها
<s> هذه هي الطريق المثلي التي يجب أن	نتخذها	

Table 4: Dual-input *n*-gram splitting of an Arabic input sentence.

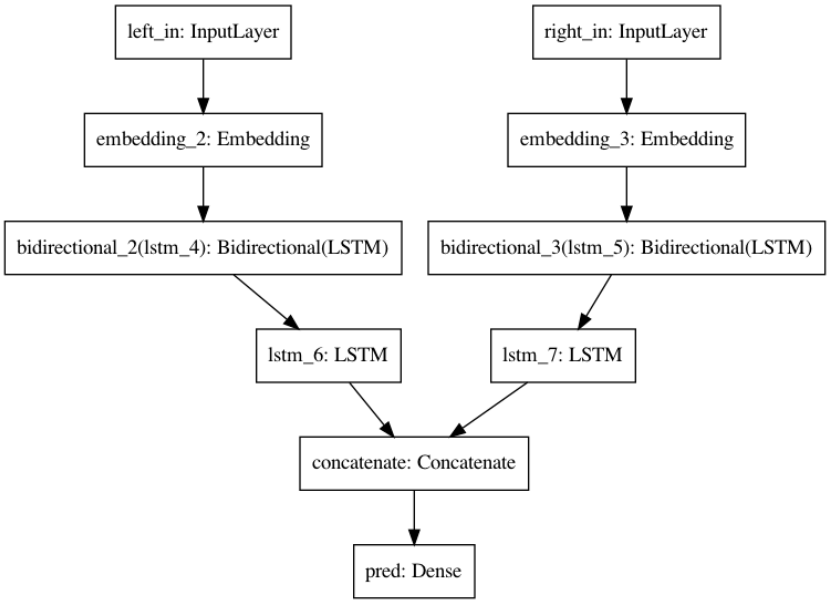


Figure 1: Dual-Input Encoder

our solution is simple, we believe its novelty lies in adding more context to the regular *many-to-one* language modelling process, which is also reflected in our results (cf. Section 3).

2.3 Inference

In the decoding process, the *many-to-one* LSTM network takes each item from the list of *n*-gram sequences generated from the input sentence (cf. Section 2.1.2) as input and predicts the next (or current) word cw. For our dual-input model, this means that we use two inputs, words before the current word $LeftW$ and words after it $RightW$. LMs are normally utilised for text generation to predict the next token or next few tokens in a sequence given the preceding tokens as context (Santhanam, 2020). Similarly, our neural network greedily decodes to search for the most likely sequences. However, in our case, instead of keeping only the 1-best candidate, we keep the *n*-best candidates B and then calculate the edit distance ed between each candidate b and the current word cw. We observed that *n* for the *n*-best list is a sensitive hyper-parameter, i.e. when we increase the size of this hyper-parameter, we obtain a better vocabulary coverage and more suggestions at the expense of many less probable candidates. In this case, the decoder may choose an incorrect word as a possible suggestion. Therefore, the value of *n* of the *n*-best list (B) is a kind of trade-off. There are three possible cases:

1. $ed = 0$: this indicates that the current word cw is found in the *n*-best candidate list B and it is likely that cw is a correct word;

2. $ed > 0$ and $ed <= 2$: this indicates that there are other suggestions for the current position in B. If the current word cw is not found at all in B or found but after several suggestions (e.g. 10), there are chances that for the current context one of these suggestions is better than cw. We also take the length of the current word cw into consideration. If the length of $cw <= 3$, we stick to $ed = 1$, and if the length of $cw > 3$, we allow $ed <= 2$. Since we have a large pool of suggestions, our current decoder uses greedy search in order to find the item in B and calculate the edit distance measure. We empirically found that this setup worked best in our case. However, in order to obtain a list of better suggestions, beam search or bidirectional beam search (Sun et al., 2017) can be applied, which has been kept for our future work;

3. if neither the current word cw nor any similar candidates are found in the *n*-best candidate list B, no output is offered.

2.3.1 Out-of-Vocabulary Tokens

There is a known limitation of neural networks, i.e. they typically operate with a fixed vocabulary. As for a more complex task such as neural machine translation (Vaswani et al., 2017), sub-word segmentation techniques such as Byte Pair Encoding (Sennrich et al., 2016) or using a unigram language model (Kudo, 2018) are usually utilised in order to solve this problem. Since we calculate the edit distance measure on tokens, it is difficult to apply sub-word segmentation or similar techniques to this problem. As far as the spelling checking is concerned, the presence of out-of-vocabulary tokens in the input sentence may cause overcorrection at decoding because they will not come as suggestions in the *n*-best list (B). In order to solve this problem, we adopted two strategies:

- handling out-of-vocabulary tokens *on-the-fly*: lemmatising long words (consisting of more than 7 characters) and comparing different lemmas to probable suggestions at the decoding time; and

- fixing the previous misspelled word before predicting cw.

We present the pseudocode of the decoding process in Algorithm 1.

Algorithm 1: Spelling Checker Algorithm

```
   // For each current word
1  for cw=1 ... CW do
       // Predict the most likely
          sequences based on the left and
          right sequences
2      B = Predict([LeftW, RightW])
3      S = []          // List of suggestions.

       // For each candidate in the n-best
          candidates B generated by the
          LM for the current word cw
4      for b=1 ... B do
           // Calculate Edit Distance ed
              between cw and b
5          if EditDistance(cw, b) = 0 then
6              break        // Word is correct
7          else if Length(cw) <= 3 AND
             EditDistance(cw, b) = 1 then
8              Add b to S
9          else if Length(cw) > 3 AND
             EditDistance(cw, b) <= 2 then
10             Add b to S
11         else
12             continue
13     Return S
```

3 Results and Discussions

This section presents the results obtained along with our findings and some discussion.

3.1 Real Word Errors (RWE)

To the best of our knowledge, there is no freely available tool that supports context-sensitive spell checking for Arabic as far as RWEs are concerned. Hence, we could not compare our proposed models with other existing spelling correction models. We obtained results to evaluate our both single-input and dual-input models on the evaluation test set, and they are reported in Table 5. Note that existing *many-to-one* spelling correction models that use LMs to detect misspelled words are in fact based on a single-input architecture, i.e. tokens before the word to be corrected used as a conditional context for correction. As mentioned earlier, our dual-input encoder takes both the preceding and following tokens in *reverse* order as the conditional context for spelling checking and correction. We see from Table 5 that both our models correctly detect the same number of RWEs. However, we can clearly see from the table that the dual-input model outperforms the single-input model in terms of the quality of suggestions and minimisation of overcorrection. We also see from Table 5 that the two strategies explained in Section 2.3.1 (i.e. comparing lemmatised variants of tokens and correcting previous words before predicting the next word) were effective in handling out-of-vocabulary words and helped minimise overcorrection.

We believe that the success of our context-sensitive approach, especially our dual-input encoding model, lies in its ability to detect RWEs regardless of the location of the word in the sentence because it takes both sides of the sentence into account for correction.

3.2 Non-Word Errors (NWE)

This section presents our results for NWEs. In this case, in addition to our single-input and dual-input models, we considered two popular Arabic spelling checkers: *LanguageTool*[9] and *Sakhr Tadqeek*.[10] We report the results obtained in Table 6. We see from the table that our dual-input model outperforms all other models in terms of quality of suggestions and minimisation of overcorrection. We also see that our models outperform *LanguageTool* and *Sakhr Tadqeek* even in terms of detecting wrong words. Additionally, we observed that while *LanguageTool* and *Sakhr Tadqeek* consider some barely-used outdated words as correct, our model detects them as potential spelling mistakes and suggests good corrections.

3.3 Prediction Examples

As mentioned in Section 2.3.1, we lemmatised those words which contain more than seven characters in order to minimise the data sparsity problem. For example, with this approach, we avoided the detection of الفعالية (*alfaaleya*) as a mistake by comparing it to other words of the same lemma such as والفعالية (*walfaaleya*) and بفعالية (*befaaleya*). Similarly, the word المصرفية (*almasrefeya*) was compared to المصرفي (*almasrefey*). We show the results obtained by applying this lemmatisation strategy to our dual-input model in Tables 5 and 6 (cf. row "*Dual-Input+Lemma*").

One of the possible ways to improve correction suggestions and avoid overcorrection is to correct the previous word (if it is a misspelled item) before predicting the next word. We observed that the collaboration of the two strategies (i.e. lemmatisation and correcting the preceding misspelled word) leads us to the best spelling detection and correction model; the evaluation scores of the best model on the test set are shown in the last row of Table 5. Note that we refer to the system that applies the first approach (correcting the previous word) as "*Dual-Input+Prev*" and the collaborative method as "*Dual-Input+Lemma+Prev*".

The last two rows of Table 5 represent the results obtained using *LanguageTool* and *Sakhr Tadqeek*. Although both tools were able to detect most NWEs, they failed to detect مواصله (*muwaseleh*) as a mistake for مواصلة (*muwaselet*).[11] This example clearly shows how such spell-checking tools may consider barely-used outdated words as real words, which are in fact spelling mistakes in the context. When it comes to correction suggestions, both *LanguageTool* and *Sakhr Tadqeek* failed to offer exact suggestions or similar alternatives for some NWEs. For example, both tools could not correct the word معدلت (*mueddelt*) as معدلات (*mueddelat*); instead, they offered words like معدلة (*mueddelet*) or معدات (*mueddat*), and *LanguageTool* offered a similar alternative معدل (*mueddel*) which is the singular form of the original word.

[9] https://languagetoolplus.com/
[10] https://tadqeek.alsharekh.org/

[11] The error comes from the letter ه which should be ة.

Model	Detected	Exact Suggestion	Similar Suggestion	Over-Correction
Single-Input	20	17	0	9
Dual-Input	20	20	N/A	5
Dual-Input+Lemma	20	20	N/A	3
Dual-Input+Prev	20	20	N/A	3
Dual-Input+Lemma+Prev	**20**	**20**	**N/A**	**1**

Table 5: Results for RWEs. The first column "Detected" represents the percentage of wrong words marked as wrong. The second column refers to the percentage of those words that are exactly found in the suggestions. The third column shows the percentage of those words whose suggestions do not include the original word but acceptable alternatives. The last column is for words marked as incorrect as they are not among the *n*-best tokens. Rows 3 and 4 represent the use of lemmatisation and previous word correction individually while row 5 shows the results of applying both methods to the dual-input model.

Model	Detected	Exact Suggestion	Similar Suggestion	Over-Correction
Single-Input	20	19	0	9
Dual-Input	20	18	1	3
Dual-Input+Lemma	20	18	1	1
Dual-Input+Prev	**20**	**19**	**0**	**0**
LanguageTool	19	11	1	3
Sakhr Tadqeek	18	14	0	0

Table 6: Results for NWEs. The first column "Detected" represents the percentage of wrong words marked as wrong. The second column refers to the percentage of those words that got the exact original word among the suggestions. The third column shows the percentage of those words whose suggestions did not include the original word but the acceptable alternatives. The last column is for words marked as incorrect as they are not among the *n*-best tokens. Rows 3 and 4 represent the use of lemmatisation and previous word correction, respectively. The last two rows show results from LanguageTool and Sakhr Tadqeek.

4 Conclusion

In this paper, we presented a deep *many-to-one* neural network-based context-sensitive spelling checking and correction model. In short, we modelled words that come both before and after the word to be corrected as the conditional context in language model predictions. The experimental results suggest that our approach has achieved considerable success in terms of both offering better correction suggestions and minimising overcorrection. Our project, *Arabisc*, code, spelling correction models and data sets are now available as an open-source project via an open repository.[12]

In the future, we plan to increase the training data size to see how our models will perform on a large-scale data set and more languages other than Arabic. The state-of-the-art bidirectional encoder representation from transformers (BERT) architecture (Devlin et al., 2018) makes use of Transformer (Vaswani et al., 2017), an attention mechanism that learns contextual relations between words in a text and can offer powerful masked language modelling. As an alternative to our LSTM LMs, we plan to investigate using BERT masked language models in *Arabisc*. We evaluated our models on a test set that contains a small number of examples. In the future, we plan to increase the size of test set examples. Currently, our models operate at word level for spell-checking and correction. This could be an issue while encountering the out-of-vocabulary items. In the future, we aim to investigate applying byte-pair encoding (Sennrich et al., 2016) or similar word-segmentation technique in our model.

[12] https://github.com/ymoslem/Arabisc

Acknowledgments

The ADAPT Centre for Digital Content Technology is funded under the Science Foundation Ireland (SFI) Research Centres Programme (Grant No. 13/RC/2106) and is co-funded under the European Regional Development Fund. The publication has emanated from research supported in part by research grants from SFI and Microsoft under Grant Numbers 13/RC/2077 and 18/CRT/6224.

References

Majed Al-Jefri and Sabri Mahmoud. 2013. Context-Sensitive Arabic Spell Checker Using Context Words and N-Gram Language Models. In *2013 Taibah University International Conference on Advances in Information Technology for the Holy Quran and Its Sciences, Madinah, Saudi Arabia*, pages 258–263.

Monojit Choudhury, Markose Thomas, Animesh Mukherjee, Anupam Basu, and Niloy Ganguly. 2007. How Difficult is it to Develop a Perfect Spell-checker? A Cross-Linguistic Analysis through Com-

plex Network Approach. In *Proceedings of the Second Workshop on TextGraphs: Graph-Based Algorithms for Natural Language Processing*, pages 81–88, Rochester, NY, USA. Association for Computational Linguistics.

Fred J. Damerau. 1964. A Technique for Computer Detection and Correction of Spelling Errors. *Commun. ACM*, 7(3):171–176.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *CoRR*, abs/1810.04805.

Matthias Hertel. 2019. *Neural Language Models for Spelling Correction*. Master's thesis, Albert-Ludwigs-Universität Freiburg im Breisgau Technische Fakultät, Freiburg, Germany.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Diederik P. Kingma and Jimmy Ba. 2015. Adam: A Method for Stochastic Optimization. In *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*.

Taku Kudo. 2018. Subword Regularization: Improving Neural Network Translation Models with Multiple Subword Candidates. *CoRR*, abs/1804.10959.

Vladimir I. Levenshtein. 1966. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics-Doklady, Vol. 10*, pages 707–710.

James L. Peterson. 1986. A Note on Undetected Typing Errors. *Commun. ACM*, 29(7):633–637.

David E Rumelhart, Geoffrey E Hinton, and Ronald J Williams. 1986. Learning representations by back-propagating errors. *Nature*, 323(6088):533.

Sivasurya Santhanam. 2020. Context based Text-generation using LSTM networks.

Rico Sennrich, Barry Haddow, and Alexandra Birch. 2016. Neural Machine Translation of Rare Words with Subword Units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany. Association for Computational Linguistics.

Qing Sun, Stefan Lee, and Dhruv Batra. 2017. Bidirectional Beam Search: Forward-Backward Inference in Neural Sequence Models for Fill-in-the-Blank Image Captioning. *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), Honolulu, Hawaii, USA*, pages 1339–1348.

Jörg Tiedemann. 2012. Parallel Data, Tools and Interfaces in OPUS. In *Proceedings of the 8th International Conference on Language Resources and Evaluation (LREC'2012)*, pages 2214–2218, Istanbul, Turkey.

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 6000–6010.

Suzan Verberne. 2002. *Context-sensitive Spell Checking Based on Word Trigram Probabilities*. Master thesis Taal, Spraak & Informatica, University of Nijmegen, Nijmegen, the Netherlands.

Tal Weiss. 2016. Deep Spelling, Rethinking Spelling Correction in the 21st Century. machinelearnings.co, accessed September 10, 2020.

Paul J Werbos. 1990. Backpropagation through time: what it does and how to do it. *Proceedings of the IEEE*, 78(10):1550–1560.

LXPER Index 2.0: Improving Text Readability Assessment for L2 English Learners in South Korea

Bruce W. Lee[1,2]
Dep. of Computer & Information Science[1]
University of Pennsylvania
PA, USA
brucelws@seas.upenn.edu

Jason Hyung-Jong Lee[2]
Research & Development Center[2]
LXPER, Inc.
Seoul, South Korea
jasonlee@lxper.com

Abstract

Developing a text readability assessment model specifically for texts in a foreign English Language Training (ELT) curriculum has never had much attention in the field of Natural Language Processing. Hence, most developed models show extremely low accuracy for L2 English texts, up to the point where not many even serve as a fair comparison. In this paper, we investigate a text readability assessment model for L2 English learners in Korea. In accordance, we improve and expand the Text Corpus of the Korean ELT curriculum (CoKEC-text). Each text is labeled with its target grade level. We train our model with CoKEC-text and significantly improve the accuracy of readability assessment for texts in the Korean ELT curriculum.

1 Introduction

Text readability assessment has been an important field of research since the 1940s. However, most research focused on the native audience in English speaking countries (Benjamin, 2012). In China, Japan, and Korea, many high and middle school students attend English language schools, in addition to their regular school classes. English subject plays an important role in the educational systems of the three countries (Mckay, 2002).

Despite the importance put in English education, the previous text readability assessment models have not been in active use in the three countries. This is due to the poor performance of traditional readability assessment models on L2 texts. We believe there is an immediate need for the development of an improved text readability assessment method for use in L2 education around the world. In this research, we put a specific focus on L2 English learners in South Korea. But our methodology is applicable to other ELT (English Language Training) curricula.

Many traditional readability assessment models are linear regression models with a small number of linguistic features, consisting of the generic features of a text like total words, total sentences, and total syllables (Kincaid et al., 1975). Such features are effective predictors of a text's readability, but more curriculum-specific features are required for L2 text readability assessments. The key distinction between native readability assessment and L2 readability assessment is that L2 students rigorously follow the specific national ELT curriculum. Unlike native students who learn English from a variety of sources, most L2 students have limited exposure to English. In this research, we reduce the average assessment error by implementing some curriculum-specific features.

The contributions of this paper are: (1) we utilize and expand CoKEC-text, one of the few graded corpora with texts from an actual L2 curriculum; (2) we investigate novel linguistic features that were rarely tested on an L2 corpus; (3) we evaluate our model against other readability models, show significantly improved accuracy, and prove that "grades" are better modeled using logistic regression, not linear regression.

2 Related Work

Research efforts in developing automated text readability assessment models for L2 students only emerged in the 2000s (Xia et al., 2016). Heilman et al. (2007) showed that grammatical features and lexical features play particularly important roles in L2 text readability prediction. Meanwhile, Vajjala and Meurers (2014) showed that the additional use of lexical features could significantly improve L2 readability assessment. Feng et al. (2010) also reported the importance of lexical features in general (for L1 speakers of English) text readability assessment.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 20–24
Suzhou, China, December 4, 2020.

However, the common limitation of the previous research in L2 readability assessment was the training corpus annotated with the grade levels for L1 readers of English. Our results, which we obtain from training our model using CoKEC-text, introduces the possibility that lexical features are not as important as the previous researchers reported. In addition, we also show that a considerably accurate text readability model can built even with a small data set if the model is optimized and the corpus is well-labeled.

3 Corpus

Since our goal is to improve the accuracy in text readability assessment of L2 texts, our ideal corpus has to fully consist of L2 texts from a non-native ELT curriculum. The base corpus that we use is CoKEC-text (Lee and Lee, 2020), which is a collection of 2760 unique grade-labeled texts that are officially administered by the Korean Ministry of Education (MOE). Similar texts are also used in the National Assessment of Educational Achievement, College Scholastic Ability Test, and MOE-approved middle school textbooks in Korea.

However, as shown in Table 1, the number of texts in the original CoKEC-text is heavily skewed to higher grades (K10 $\sim$ K12.5) than in lower grades (K7 $\sim$ K9). Such a disparity can affect the accuracy of our regression results and can become troublesome in predicting the lower grade texts' readability. Thus, we decided to collect about 900 more texts from Korean MOE-approved middle school textbooks and use them to create an expanded version of CoKEC-text.

In addition, we found some K7 $\sim$ K9 texts that are only partially English. They contained ASCII Korean Characters (often explanations of difficult English words by the author). These count as a token (or possibly, even a sentence from case to case) in the NLTK parsing process but provide no meaningful linguistic properties. This can produce miscalculations of the "average of x" (e.g., the average number of words per sentence) features that we discuss in Section 4. We manually went through every text to make sure that clean data is used for model training. Our final training corpus consists of 3700 original L2 texts from K7 $\sim$ K12.5. K12.5 grade texts are from CSAT, which is a college entrance exam for Korean universities. In general, the Korean grades K7 to K12 are for middle and high school students of ages 13 to 19.

Difficulty Labels	Original	Expanded
12.5	691	691
12	590	601
11	596	602
10	571	580
9	80	313
8	215	302
7	17	305

Table 1: Number of texts in two corpus versions

4 Selecting features

We now describe the 35 linguistic features we studied. Table 2 contains a list of the features with a shortcode name used throughout this paper. The list is divided into five parts: traditional features, POS-based features, entity density features, lexical chain features, and word difficulty features.

4.1 Traditional Features

We first implemented some traditional features from the popular Flesch-Kincaid model: aWPS (average number of Words per Sentence), aSPW (average number of Syllables per Word), and P3T (words with more than 3 syllables per Text). These are one of the earliest linguistic features studied in text readability prediction, but they prove to be still useful in recent studies (Feng et al., 2010).

4.2 POS-based Features

A number of researchers commonly reported that POS-based features are effective in text readability prediction. In particular, Peterson and Ostendorf (2009) investigated the following features: aNP, aNN, aVP, aAdj, aSBr, aPP, nNP, nNN, nVP, nAdj, aSBr, nPP. Lee and Lee (2020) proved that these features are highly correlated with the difficulty of L2 texts. However, their dataset mostly consisted of K10, K11, K12 texts, and their evaluation was conducted only on K9 $\sim$ K12. Thus, there exists a possibility that the result was heavily influenced by higher grade L2 texts. We evaluate these features again with our expanded version of CoKEC-text.

4.3 Entity Density Features

We implement entity density features in an attempt to account for the difficulty in comprehending conceptual information in texts. Such information is often introduced by entities, or more specifically, general nouns and named entities.

The density of entities introduced in a text relates to the working memory burden, which has an increasing trend in a positive correlation with the age of the reader. Our main task is to develop a model that would be particularly useful to L2 student groups, and accurately classify the given texts to the respective student grade level. Hence, we believe that these entity density features are great predictors. Some of these features were never tested on an L2 corpus, and the results we obtain are novel.

4.4 Lexical Chain Features

We believe that the accuracy of L2 text readability can be improved by incorporating lexical chain features as well. Since L2 readers have limited exposure to English compared to native readers, we hypothesize that L2 readers work harder in connecting several entities and recognizing the semantic relationship. Entities that form these semantic relations are connected throughout the text in the form of lexical chains. However, in Table 3 we observe that lexical chain features are weakly correlated to target grade levels of L2 texts in Korea.

4.5 Word Difficulty Features

Native English readers learn vocabulary from a variety of sources. On the other hand, most L2 students learn new English words step by step, following the respective national ELT curriculum. Hence, implementing curriculum specific features related to vocabularies can be particularly useful in predicting the text difficulty for L2 students.

We use CoKEC-word to identify the difficulty of words (Lee and Lee, 2020). The word corpus is a classification of 30608 words in 6 levels. It only consists of the words that previously appeared in the Korean ELT curriculum. We focused on the vocabularies in levels B, C, D, E, and F. This covers vocabularies from K5 to college level.

4.6 Parsing and Counting Modules

We used a combination of spaCy (popular open-source library for NLP) (Honnibal and Montani, 2017), NLTK (NLP toolkit for Python) (Bird et al., 2009), Gensim (famous for topic modeling and FastText model) (Řehůřek and Sojka, 2010), and the Berkeley Neural Parser (constituency parser) (Kitaev and Klein, 2018) to parse and count the features described in this section.

Code	Feature Description
aWS	average number of Words per sent
aSPW	avg num of Syllables per word
P3T	% of words with $\geq$ to 3 syl
nWD	total number of Words per Doc
aNP	avg num of Noun Phrases per sent
aNN	avg num of proper nouns per sent
aVP	avg num of Verb Phrases per sent
aAdj	avg num of Adjectives per sent
aSBr	avg num of Subord. Clauses per sent
aPP	avg num of Prepos. Phrases per sent
nNP	total num of Noun Phrases per sent
nNN	total num of proper nouns per sent
nVP	total num of Verb Phrases per sent
nAdj	total number of Adjectives per sent
nSBr	total num of Subord. Clauses per sent
nPP	total num of Prepos. Phrases per sent
PND	% of named entities per doc
PNS	% of named entities per sent
nUE	total number of Unique Entities
aEM	avg num of Entity Mentions per sent
aUE	avg num of Unique Entities per sent
nLC	total num of Lexical Chains
aLCW	avg num of Lexical Chains per word
aLCS	avg num of Lex Chains per noun sent
aLCN	avg num of Lex Chains per noun phrase
nBw	total num of lev B (K5-8) words
aBw	avg num of lev B words per word
nCw	total num of lev C (K8-9) words
aCw	avg num of lev C words per word
nDw	total num of lev D (K9-11) words
aDw	avg num of lev D words per word
nEw	total num of lev E (K11-12) words
aEw	avg num of lev E words per word
nEw	total num of lev F (college) words
aFw	avg num of lev F words per word

Table 2: Number of texts in two corpus versions

To keep operation simple, only NLTK and Berkeley Neural Parser were used in the previous version of LXPER Index. However, our further investigation show that certain tasks are performed at much higher accuracy by complementary libraries. For example, spaCy showed the highest accuracy at recognizing a sentence, and Gensim improved the lexical chaining process.

4.7 Selecting Features

We computed the Pearson correlation value of each feature and checked if it was significant enough (correlation $>$ 0.07) in predicting the target grade

level of a text. The "Cor" column in Table 3 lists the correlation value of each feature. We ordered the list in decreasing correlation values. Next, we removed the features that are highly correlated ("Paired?" column). The "Include?" column in Table 3 summarizes the final features.

Code	Cor	Sig?	Paired?	Include?
nDw	0.532	Y	Y	Y
aWS	0.512	Y	N	Y
aSPW	0.499	Y	N	Y
aDw	0.487	Y	Y	N
nBw	0.454	Y	Y	Y
aNP	0.446	Y	N	Y
P3T	0.444	Y	N	Y
aNN	0.434	Y	N	Y
aPP	0.423	Y	N	Y
nPP	0.417	Y	N	Y
nCw	0.402	Y	Y	Y
nEw	0.399	Y	Y	Y
nAdj	0.394	Y	N	Y
aAdj	0.378	Y	N	Y
nNN	0.376	Y	N	Y
aVP	0.323	Y	N	Y
nWD	0.321	Y	N	Y
nNP	0.308	Y	N	Y
aSBr	0.298	Y	N	Y
aCw	0.289	Y	Y	N
aBw	0.274	Y	Y	N
nSBr	0.221	Y	N	Y
aEw	0.221	Y	Y	Y
nLC	0.212	Y	N	Y
PND	0.201	Y	N	Y
nEw	0.195	Y	Y	Y
PNS	0.174	Y	N	Y
aLCW	0.154	Y	N	Y
nVP	0.126	Y	N	Y
aLCN	0.0995	Y	N	Y
aFw	0.0976	Y	Y	N
aLCS	0.0913	Y	N	Y
aUE	0.0884	Y	N	Y
aEM	0.0792	N	N	N
nUE	0.00833	N	N	N

Table 3: Selecting features

5 Readability Assessment

We built a logistic regression model and trained it with the new expanded version of CoKEC-text to complete our assessment tool; our model is programmed in Python. To evaluate the new model's effectiveness for L2 students in Korea, we prepared a separate test corpus. The first part (K10 $\sim$ K12) of our test corpus is from the official mock tests that were used by KICE (Korea Institute of Curriculum & Evaluation) to assess the educational achievement of high school students from 2017 to 2020. There are 270 texts in the first part of our test corpus (K10: 90 texts, K11: 90 texts, K12: 90 texts). The second part of our corpus is from the government-approved middle school textbooks (K7: 90 texts, K8: 90 texts, K9: 90 texts).

Type	F-K	D-C	LX 1.0	LX 2.0
K7	4.89	5.38	9.21	7.3
K8	5.44	5.02	9.43	8.45
K9	5.78	5.53	9.86	9.04
K10	10.6	7.95	10.9	10.5
K11	9.66	7.57	11.4	11.3
K12	9.21	7.31	11.5	11.6
Avg Er. (in K)	2.10	3.04	1.05	0.34

Table 4: Final results

We collected the texts from two sources to test how our readability assessment model performs on different types of texts. Ideally, our results should show a continuous increase from K7 to K12 texts. Our target average assessment error is below 0.5 grade level. Table 4 summarizes our results. We compare our LXPER Index 2.0 (LX 2.0) to traditionally popular models like Flesch-Kincaid (F-K) (Kincaid et al., 1975), Dale-Chall (D-C) (Dale and Chall, 1949), and the previous LXPER Index 1.0 (LX 1.0) (Lee and Lee, 2020).

We also wanted to compare our model to the more recently developed models, but we could not find any suitable L2 readability index. We attempted comparison with Lexile Score and CohMetrix L2 Readability Score (Crossley et al., 2008). However, the models had a completely different grading scale and did not show a consistently increasing trend with grades. This was also reported in our previous research (Lee and Lee, 2020).

6 Conclusion

In this research, we introduced LXPER Index 2.0, a readability assessment tool that incorporates traditional, POS, entity density, lexical chain, and word difficulty features. Then, we trained the model on our own expanded version of CoKEC-text. We obtained a continuously increasing output for L2 texts from K7 to K12. In addition, we achieved our initial target average accuracy error of less than 0.5

grade levels, which is more accurate than any L2 text readability prediction model we are aware of.

The improvements we report in this paper are largely due to two changes: 1. the CoKEC expansion and 2. the use of a logistic regression model. The contribution from the corpus is quite obvious in that our model could now learn more about the lower grades (K7 $\sim$ K9). However, the contribution from the change of the regression model is something that we should put more thought into. But it seems evident that the "grades" classification task is better modeled with a logistic regression model. A possible explanation could be that the difficulty of a text does not linearly correlate with the target grades.

Even though we wanted to test our model on other East Asia L2 ELT curricula, like Japan and China, we could not implement due to the lack of openly-available corpus in the countries. The novelty of the LXPER Index model is that it focuses on in-curriculum text readability analysis, possibly even with a small data set of less than 4000 texts. Thus, applying the model will fail to give meaningful outcomes without a pre-processed and labeled corpus like CoKEC. Thus, the application of a similar model on those countries would first require foundation research on constructing corpora.

References

Rebekah G. Benjamin. 2012. Reconstructing readability: Recent developments and recommendations in the analysis of text difficulty. *Educational Psychology Review*, 24(21).

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Scott A. Crossley, Jerry Greenfield, and Danielle S. McNamara. 2008. Assessing text readability using cognitively based indices. *TESOL Quarterly*, 42(3).

Edgar Dale and Jeanne S. Chall. 1949. The concept of readability. *Elementary English*, 26(23).

Lijun Feng, Martin Jansche, Matt Huenerfauth, and Noemie Elhadad. 2010. A comparison of features for automatic readability assessment. In *Proceedings of the 23rd International Conference on Computational Linguistics*, pages 276–284.

Michael Heilman, Kevyn Collins-Thompson, Jamie Callan, and Maxine Eskenazi. 2007. Combining lexical and grammatical features to improve readability measures for first and second language text. In *Proceedings of North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 460–467.

Matthew Honnibal and Ines Montani. 2017. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. Version2.

J. Peter Kincaid, Robert P. Fishburne Jr., Richard L. Rogers, and Brad S. Chissom. 1975. Derivation of new readability formulas (automated readability index, fog count, and flesch reading ease formula) for navy enlisted personnel. *Research Branch Report*, pages 8–75.

Nikita Kitaev and Dan Klein. 2018. Constituency parsing with a self-attentive encoder. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, pages 2676—-2686.

Bruce W. Lee and Jason H. Lee. 2020. Lxper index: A curriculum-specific text readability assessment model for efl students in korea. *International Journal of Advanced Computer Science and Applications*, 11(8).

Sandra L. Mckay. 2002. *Teaching English as an International Language: Rethinking Goals and Perspectives*. OUP Oxford.

Sarah E. Peterson and Mari Ostendorf. 2009. A machine learning approach to reading level assessment. *Computer Speech and Language*, 23.

Sowmya Vajjala and Detmar Meurers. 2014. Readability assessment for text simplification: From analyzing documents to identifying sentential simplification. *International Journal of Applied Linguistics*, 165(2).

Menglin Xia, Ekaterina Kochmar, and Ted Briscoe. 2016. Text readability assessment for second language learners. In *Proceedings of the 11th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 12–22.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of LREC Workshop on New Challenges for NLP Frameworks*, pages 45–50.

Overview of NLPTEA-2020 Shared Task for Chinese Grammatical Error Diagnosis

Gaoqi Rao Erhong Yang Baolin Zhang
Beijing Language and Culture University
{raogaoqi, yangerhong, zhangbaolin}@blcu.edu.cn

Abstract

This paper presents the NLPTEA 2020 shared task for Chinese Grammatical Error Diagnosis (CGED) which seeks to identify grammatical error types, their range of occurrence and recommended corrections within sentences written by learners of Chinese as a foreign language. We describe the task definition, data preparation, performance metrics, and evaluation results. Of the 30 teams registered for this shared task, 17 teams developed the system and submitted a total of 43 runs. System performances achieved a significant progress, reaching F1 of 91% in detection level, 40% in position level and 28% in correction level. All data sets with gold standards and scoring scripts are made publicly available to researchers.

1 Introduction

Automated grammar checking for learners of English as a foreign language has achieved obvious progress. Helping Our Own (HOO) is a series of shared tasks in correcting textual errors (Dale and Kilgarriff, 2011; Dale et al., 2012). The shared tasks at CoNLL 2013 and 2014 focused on grammatical error correction, increasing the visibility of educational application research in the NLP community (Ng et al., 2013; 2014).

Many of these learning technologies focus on learners of English as a Foreign Language (EFL), while relatively few grammar checking applications have been developed to support Chinese as a Foreign Language (CFL) learners. Those applications which do exist rely on a range of techniques, such as statistical learning (Chang et al, 2012; Wu et al, 2010; Yu and Chen, 2012), rule-based analysis (Lee et al., 2013), neuro network modelling (Zheng et al., 2016; Fu et al., 2018) and hybrid methods (Lee et al., 2014; Zhou et al., 2017).

In response to the limited availability of CFL learner data for machine learning and linguistic analysis, the ICCE-2014 workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA) organized a shared task on diagnosing grammatical errors for CFL (Yu et al., 2014). A second version of this shared task in NLP-TEA was collocated with the ACL-IJCNLP-2015 (Lee et al., 2015), COLING-2016 (Lee et al., 2016). Its name was fixed from then on: Chinese Grammatical Error Diagnosis (CGED). As a part of IJCNLP 2017, the shared task was organized (Rao et al., 2017). In conjunction with NLP-TEA workshop in ACL 2018, CGED was organized again (Rao et al., 2018). The main purpose of these shared tasks is to provide a common setting so that researchers who approach the tasks using different linguistic factors and computational techniques can compare their results. Such technical evaluations allow researchers to exchange their experiences to advance the field and eventually develop optimal solutions to this shared task.

The rest of this paper is organized as follows. Section 2 describes the task in detail. Section 3 introduces the constructed data sets. Section 4 proposes evaluation metrics. Section 5 reports the results of the participants' approaches. Conclusions are finally drawn in Section 6.

2 Task Description

The goal of this shared task is to develop NLP techniques to automatically diagnose (and furtherly correct) grammatical errors in Chinese sentences written by CFL learners. Such errors are

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 25–35
Suzhou, China, December 4, 2020.

defined as PADS: redundant words (denoted as a capital "R"), missing words ("M"), word selection errors ("S"), and word ordering errors ("W"). The input sentence may contain one or more such errors. The developed system should indicate which error types are embedded in the given unit (containing 1 to 5 sentences) and the position at which they occur. Each input unit is given a unique number "sid". If the inputs contain no grammatical errors, the system should return: "sid, correct". If an input unit contains the grammatical errors, the output format should include four items "sid, start_off, end_off, error_type", where start_off and end_off respectively denote the positions of starting and ending character at which the grammatical error occurs, and error_type should be one of the defined errors: "R", "M", "S", and "W". Each character or punctuation mark occupies 1 space for counting positions. Example sentences and corresponding notes are shown as Table 1 shows. This year, we only have one track of HSK.

Hanyu Shuiping Kaoshi (HSK)
Example 1 Input: (sid=00038800481)　我根本不能了解这妇女辞职回家的现象。在这个时代，为什么放弃自己的工作，就回家当家庭主妇？ Output: 00038800481, 6, 7, S 00038800481, 8, 8, R (Notes: "了解"should be "理解". In addition, "这" is a redundant word.) Example 2 Input: (sid=00038800464)我真不明白。她们可能是追求一些前代的浪漫。 Output: 00038800464, correct Example 3 Input: (sid=00038801261)人战胜了饥饿，才努力为了下一代作更好的、更健康的东西。 Output: 00038801261, 9, 9, M 00038801261, 16, 16, S (Notes: "能" is missing. The word "作"should be "做". The correct sentence is "才能努力为了下一代做更好的") Example 4 Input: (sid=00038801320)饥饿的问题也是应该解决的。世界上每天由于饥饿很多人死亡。 Output: 00038801320, 19, 25, W (Notes: "由于饥饿很多人" should be "很多人由于饥饿")

Table 1: Example sentences and corresponding notes

3 Data Sets

The learner corpora used in our shared task were taken from the writing section of the HSK (Pinyin of *Hanyu Shuiping Kaoshi*, Test of Chinese Level) (Cui et al, 2011; Zhang et al, 2013).

Native Chinese speakers were trained to manually annotate grammatical errors and provide corrections corresponding to each error. The data were then split into two mutually exclusive sets as follows.

(1) Training Set: All units in this set were used to train the grammatical error diagnostic systems. Each unit contains 1 to 5 sentences with annotated grammatical errors and their corresponding corrections. All units are represented in SGML format, as shown in Table 2. We provide 1129 training units with a total of 2,909 grammatical errors, categorized as redundant (678 instances), missing (801), word selection (1228) and word ordering (201).

In addition to the data sets provided, participating research teams were allowed to use other public data for system development and implementation. Use of other data should be specified in the final system report.

#Units	**#Correct**	**#Erroneous**
1,457 (100%)	307 (21.07%)	1,150 (78.93%)

Table 3: The statistics of correct sentences in testing set.

Test Set: This set consists of testing units used for evaluating system performance. Table 3 shows statistics for the testing set for this year. According to the sampling in the writing sessions in HSK, over 40% of the sentences contain no error. This was simulated in the test set, in order to test the performance of the systems in false positive identification. The distributions of error types (Table 4) are similar with that of the training set. The proportion of the correct sentences is sampled from data of the online Dynamic Corpus of HSK[1].

Error Type	
#R	769 (21.05%)
#M	864 (23.65%)
#S	1694 (46.36%)
#W	327 (8.95%)
#Error	3,654 (100%)

Table 4: The distributions of error types in testing set.

4 Performance Metrics

Table 5 shows the confusion matrix used for evaluating system performance. In this matrix, TP (True Positive) is the number of sentences with grammatical errors are correctly identified by the developed system; FP (False Positive) is the number of sentences in which non-existent grammatical errors are identified as errors; TN (True Negative) is the number of sentences without grammatical errors that are correctly identified as such; FN (False Negative) is the number of sentences with grammatical errors which the system incorrectly identifies as being correct.

The criteria for judging correctness are determined at three levels as follows.

(1) Detection-level: Binary classification of a given sentence, that is, correct or incorrect, should be completely identical with the gold standard. All error types will be regarded as incorrect.

(2) Identification-level: This level could be considered as a multi-class categorization problem. All error types should be clearly identified. A correct case should be completely identical with the gold standard of the given error type.

(3) Position-level: In addition to identifying the error types, this level also judges the occurrence range of the grammatical error. That is to say, the system results should be perfectly identical with the quadruples of the gold standard.

Besides the traditional criteria in the past share tasks, Correction-level was introduced to CGED since 2018.

(4) Correction-level: For the error types of Selection and Missing, recommended corrections are required. At most 3 recommended corrections are allowed for each S and M type error. In this level the amount of the corrections recommended would influence the precision and F1 in this level. The trust of the recommendation would be test. The sub-track TOP1 count only one recommended correction, while TOP3 count one hit, if one correction in three hits the golden standard, ignoring its ranking.

The following metrics are measured at all levels with the help of the confusion matrix.

- False Positive Rate = FP / (FP+TN)
- Accuracy = (TP+TN) / (TP+FP+TN+FN)
- Precision = TP / (TP+FP)
- Recall = TP / (TP+FN)
- F1 =2*Precision*Recall / (Precision + Recall)

For example, for 4 testing inputs with gold standards shown as “00038800481, 6, 7, S”, “00038800481, 8, 8, R”, “00038800464, correct”, “00038801261, 9, 9, M”, “00038801261, 16, 16, S” and “00038801320, 19, 25, W”, the system may output the result as “00038800481, 2, 3, S”, “00038800481, 4, 5, S”, “00038800481, 8, 8, R”, “00038800464, correct”, “00038801261, 9, 9, M”, “00038801261, 16, 19, S” and “00038801320, 19, 25, M”. The scoring script will yield the following performance.

False Positive Rate (FPR) = 0 (=0/1)
Detection-level: Precision = 1 (=3/3)
Recall = 1 (=3/3)
F1 = 1 (=(2*1*1)/(1+1))
Identification-level: Precision = 0.8 (=4/5)
Recall = 0.8 (=4/5)
F1 = 0.8 (=(2*0.8*0.8)/(0.8+08))
Position-level: Precision = 0.3333 (=2/6)
Recall = 0.4 (=2/5)
F1 = 0.3636 (=(2*0.3333*0.4)/(0.3333+0.4))

[1] http://bcc.blcu.edu.cn/hsk

<DOC>
<TEXT id="200307109523200140_2_2x3">
因为养农作物时不用农药的话，生产率较低。那肯定价格要上升，那有钱的人想吃多少，就吃多少。左边的文中已提出了世界上的有几亿人因缺少粮食而挨饿。
</TEXT>
<CORRECTION>
因为种植农作物时不用农药的话，生产率较低。那价格肯定要上升，那有钱的人想吃多少，就吃多少。左边的文中已提出了世界上有几亿人因缺少粮食而挨饿。
</CORRECTION>
<ERROR start_off="3" end_off="3" type="S"></ERROR>
<ERROR start_off="22" end_off="25" type="W"></ERROR>
<ERROR start_off="57" end_off="57" type="R"></ERROR>
</DOC>

<DOC>
<TEXT id="200210543634250003_2_1x3">
对于“安乐死”的看法，向来都是一个极具争议性的题目，因为毕竟每个人对于死亡的观念都不一样，怎样的情况下去判断，也自然产生出很多主观和客观的理论。每个人都有着生存的权利，也代表着每个人都能去决定如何结束自己的生命的权利。在我的个人观点中，如果一个长期受着病魔折磨的人，会是十分痛苦的事，不仅是病人本身，以致病者的家人和朋友，都是一件难受的事。
</TEXT>
<CORRECTION>
对于“安乐死”的看法，向来都是一个极具争议性的题目，因为毕竟每个人对于死亡的观念都不一样，无论在怎样的情况下去判断，都自然产生出很多主观和客观的理论。每个人都有着生存的权利，也代表着每个人都能去决定如何结束自己的生命。在我的个人观点中，如果一个长期受着病魔折磨的人活着，会是十分痛苦的事，不仅是病人本身，对于病者的家人和朋友，都是一件难受的事。
</CORRECTION>
<ERROR start_off="46" end_off="46" type="M"></ERROR>
<ERROR start_off="56" end_off="56" type="S"></ERROR>
<ERROR start_off="106" end_off="108" type="R"></ERROR>
<ERROR start_off="133" end_off="133" type="M"></ERROR>
<ERROR start_off="151" end_off="152" type="S"></ERROR>
</DOC>

Table 2: A training sentence denoted in SGML format.

Confusion Matrix		System Results	
		Positive (Erroneous)	Negative (Correct)
Gold Standard	Positive	TP (True Positive)	FN (False Negative)
	Negative	FP (False Positive)	TN (True Negative)

Table 5: Confusion matrix for evaluation.

5 Evaluation Results

Table 6 summarizes the submission statistics for the 17 participating teams. In the official testing phase, each participating team was allowed to submit at most three runs. Of the 17 teams, 11 teams submitted their testing results in Correction-level, for a total of 43 runs.

Participant (Ordered by names)	**#Runs**	**Correction-level**

Boli	2	√
CYUT	2	-
DumbCat	1	√
Flying	3	√
LDU	3	-
NJU-NLP	3	-
OrangePlus	3	√
PCJG	3	√
SDU_MLA	1	-
SPPD	3	-
TextCC-CloudPoineer	3	√
TMU-NLP	1	√
UNIPUS-Flaubert	3	√
XHJZ	3	√
YD_NLP	3	√
ZZUNLP-HAN	3	√
ZZUNLP-YAN	3	-

Table 6: Submission statistics for all participants.

Table 7 to 11 show the testing results of the CGED2020 in 6 tracks: false positive rate (FPR), detection level, identification level, position level and correction level (in two settings: top1 and top3). All runs of top F1 score are highlighted in the tables. The CYUT achieved the lowest FPR of 0.0163, about one third of the lowest FPR in the CGED 2018. Detection-level evaluations are designed to detect whether a sentence contains grammatical errors or not. A neutral baseline can be easily achieved by reporting all testing sentences containing errors. According to the test data distribution, the baseline system can achieve an accuracy of 0.7893. However, not all systems performed above the baseline. The system result submitted by NJU-NLP achieved the best detection F1 of 0.9122, beating the 0.9 mark for the first time. For identification-level evaluations, the systems need to identify the error types in a given unit. The system developed by Flying and OrangePlus provided the highest F1 score of 0.6736 and 0.6726 for grammatical error identification. For position-level, Flying achieved the best F1 score of 0.4041, crossing the 0.4 mark for the first time. OrangePlus reached 0.394. Perfectly identifying the error types and their corresponding positions is difficult because the error propagation is serious. In correction-level, UNIPUS-Flaubert achieved best F1 of 0.1891 in top1 setting and YD_NLP of 0.1885 top3 setting.

In CGED 2020, the implementation of pre-trained model like BERT achieved significant improvement in many tracks. The "standard pipe-line" biLSTM+CRF in CGED2017 and 2018 is replaced. Hybrid methods based on pre-trained model were proposed by most of the teams. ResNet, graph convolution network and data argumentation appeared for the first time in the solutions. The rethinking the data construction (including pseudo data generation) and feature selection did not attract the attention of the participants. However, the balance of the FPR and other track did not progress a lot. The rough merging strategies implemented in hybrid methods and the over generation of generation models may lead the drop in FPR. From organizers' perspectives, a good system should have a high F1 score and a low false positive rate.

In summary, none of the submitted systems provided a comprehensive superior performance using different metrics, indicating the difficulty of developing systems for effective grammatical error diagnosis, especially in CFL contexts. It is worth noting that in the track of detection, the performance over 0.9 is close to the application of actual scene. In the highly focused track of position and correction, variant teams lead the ranks, unlike the past CGEDs. It's a very exciting phenomena indicating the attraction the task increased quickly.

TEAM Name	Run	FPR	TEAM Name	Run	FPR

Boli	1	0.7590	SPPD	1	0.1498
	2	0.7687		2	0.1107
CYUT	1	**0.0163**		3	0.0749
	2	0.5472	TextCC-CloudPoineer	1	0.2476
DumbCat	1	0.2052		2	0.2834
Flying	1	0.1010		3	0.4104
	2	0.2573	TMU-NLP	1	0.1726
	3	0.3257	UNIPUS-Flaubert	1	0.2508
LDU	1	0.0423		2	0.2443
	2	0.0489		3	0.4756
	3	0.0391	XHJZ	1	0.8762
NJU-NLP	1	0.6124		2	0.7752
	2	0.2378		3	0.7068
	3	0.0554	YD_NLP	1	0.2052
OrangePlus	1	0.2443		2	0.2345
	2	0.2964		3	0.2182
	3	0.2606	ZZUNLP-HAN	1	0.6645
PCJG	1	0.5440		2	0.6775
	2	0.8176		3	0.7394
	3	0.3844	ZZUNLP-YAN	1	0.8078
SDU_MLA	1	0.5179		2	0.7557
				3	0.6938

Table7. Results of CGED 2020 in False Positive Rate (FPR)

TEAM Name	**RUN**	**Detection Level**			**TEAM Name**	**RUN**	**Detection Level**		
		Pre.	**Rec.**	**F1**			**Pre.**	**Rec.**	**F1**
Boli	1	0.8149	0.8922	0.8518	SPPD	1	0.9541	0.8313	0.8885
	2	0.814	0.8983	0.8541		2	0.9649	0.8139	0.8830
CYUT	1	0.9875	0.3443	0.5106		3	0.9743	0.7574	0.8523
	2	0.8117	0.6296	0.7091	TextCC-CloudPoineer	1	0.9265	0.7565	0.8329
DumbCat	1	0.9078	0.5391	0.6765		2	0.9182	0.7809	0.8440
Flying	1	0.9649	0.7409	0.8382		3	0.8784	0.7913	0.8326
	2	0.9273	0.6213	0.6736	TMU-NLP	1	0.9404	0.7270	0.8200
	3	0.9101	0.8800	**0.8948**	UNIPUS-Flaubert	1	0.9214	0.7852	0.8479
LDU	1	0.9851	0.7496	0.8514		2	0.9207	0.7574	0.8311
	2	0.9828	0.7452	0.8477		3	0.8782	0.9157	**0.8966**
	3	0.9851	0.6887	0.8106	XHJZ	1	0.8062	0.9730	0.8818
NJU-NLP	1	0.8565	0.9757	**0.9122**		2	0.8069	0.5874	0.6799
	2	0.9303	0.8478	0.8872		3	0.8180	0.8478	0.8326
	3	0.9739	0.5513	0.7041	YD_NLP	1	0.9387	0.8383	0.8857
OrangePlus	1	0.9282	0.8435	0.8838		2	0.9319	0.8565	**0.8926**
	2	0.9161	0.8643	0.8895		3	0.9357	0.8478	0.8896
	3	0.9252	0.8600	**0.8914**	ZZUNLP-HAN	1	0.8262	0.8435	0.8348
PCJG	1	0.8225	0.6730	0.7403		2	0.8145	0.7939	0.8041
	2	0.8142	0.9565	0.8796		3	0.8136	0.8617	0.8370

	3	0.8698	0.6852	0.7665		1	0.8118	0.9304	0.8671
SDU_MLA	1	0.8138	0.5965	0.6884	ZZUNLP-YAN	2	0.8182	0.9078	0.8607
						3	0.8254	0.8757	0.8498

Table8. Results of CGED 2020 in Detection Level

TEAM Name	RUN	Identification Level			TEAM Name	RUN	Identification Level		
		Pre.	Rec.	F1			Pre.	Rec.	F1
Boli	1	0.5883	0.5347	0.5602	SPPD	1	0.7166	0.5892	0.6467
	2	0.5872	0.5389	0.5620		2	0.7600	0.5676	0.6499
CYUT	1	0.6412	0.166	0.2637		3	0.7843	0.4862	0.6003
	2	0.4902	0.2768	0.3538	TextCC-CloudPoineer	1	0.7090	0.4982	0.5852
DumbCat	1	0.7002	0.3929	0.5034		2	0.7034	0.5285	0.6035
Flying	1	0.7769	0.4738	0.5886		3	0.6751	0.5051	0.5779
	2	0.7356	0.6213	**0.6736**	TMU-NLP	1	0.6980	0.4228	0.5266
	3	0.7320	0.6011	0.6601	UNIPUS-Flaubert	1	0.7415	0.4890	0.5893
LDU	1	0.5714	0.6897	0.6250		2	0.7515	0.4710	0.5791
	2	0.5715	0.6874	0.6241		3	0.6507	0.6420	0.6463
	3	0.75	0.2772	0.4048	XHJZ	1	0.5669	0.6714	0.6147
NJU-NLP	1	0.5571	0.8432	0.6709		2	0.5897	0.6011	0.5953
	2	0.7018	0.5779	0.6339		3	0.6063	0.5873	0.5966
	3	0.7939	0.2975	0.4328	YD_NLP	1	0.7788	0.5503	0.6449
OrangePlus	1	0.7223	0.6121	0.6627		2	0.7623	0.5678	0.6508
	2	0.7188	0.5450	0.6200		3	0.7711	0.5577	0.6473
	3	0.7230	0.6287	**0.6726**	ZZUNLP-HAN	1	0.5856	0.4416	0.5035
PCJG	1	0.6136	0.3154	0.4166		2	0.5053	0.4127	0.4543
	2	0.5926	0.5678	0.5799		3	0.5018	0.5060	0.5039
	3	0.6499	0.3687	0.4705	ZZUNLP-YAN	1	0.5899	0.5126	0.5485
SDU_MLA	1	0.5411	0.2813	0.3701		2	0.6150	0.5076	0.5562
						3	0.64	0.5214	0.5746

Table9. Results of CGED 2020 in Identification Level

TEAM Name	RUN	Position Level			TEAM Name	RUN	Position Level		
		Pre.	Rec.	F1			Pre.	Rec.	F1
Boli	1	0.2284	0.1719	0.1962	SPPD	1	0.3595	0.2671	0.3065
	2	0.2284	0.1755	0.1985		2	0.4225	0.2822	0.3384
CYUT	1	0.0134	0.0033	0.0053		3	0.4673	0.2466	0.3228
	2	0.0136	0.0068	0.0091	TextCC-CloudPoineer	1	0.3612	0.2392	0.2878
DumbCat	1	0.3565	0.1828	0.2417		2	0.3518	0.2518	0.2935
Flying	1	0.4970	0.2529	0.3352		3	0.3577	0.2318	0.2813
	2	0.4320	0.3514	0.3876	TMU-NLP	1	0.3460	0.1639	0.2224
	3	0.4715	0.3536	**0.4041**	UNIPUS-Flaubert	1	0.4758	0.2343	0.3140
LDU	1	0.1397	0.1612	0.1497		2	0.4606	0.2288	0.3057
	2	0.1407	0.1621	0.1506		3	0.3147	0.2739	0.2929

	3	0	0	0.0000	XHJZ	1	0.2368	0.2849	0.2586
NJU-NLP	1	0.2097	0.4648	0.2890		2	0.2610	0.2663	0.2636
	2	0.4008	0.288	0.3351		3	0.2993	0.2655	0.2814
	3	0.5757	0.1519	0.2404	YD_NLP	1	0.5145	0.2965	0.3762
OrangePlus	1	0.4366	0.3372	0.3805		2	0.4822	0.3011	0.3707
	2	0.4241	0.2731	0.3323		3	0.5011	0.2995	0.3749
	3	0.4428	0.361	**0.3977**	ZZUNLP-HAN	1	0.2502	0.1472	0.1854
PCJG	1	0.0885	0.0342	0.0494		2	0.0996	0.0665	0.0798
	2	0.2582	0.2143	0.2342		3	0.067	0.0613	0.0640
	3	0.3282	0.1399	0.1962	ZZUNLP-YAN	1	0.29	0.1941	0.2326
SDU_MLA	1	0.0708	0.0276	0.0398		2	0.2874	0.1892	0.2282
						3	0.2783	0.2042	0.2356

Table10. Results of CGED 2020 in Position Level

TEAM Name	**RUN**	**Correction Level（TOP1）**			**Correction Level（TOP3）**		
		Pre.	**Rec.**	**F1**	**Pre.**	**Rec.**	**F1**
Boli	1	0.079	0.0629	0.0700	0.079	0.0629	0.0700
	2	0.0768	0.0629	0.0692	0.0768	0.0629	0.0692
DumbCat	1	0.2502	0.1126	0.1553	0.2502	0.1126	0.1553
Flying	1	0.246	0.1149	0.1567	0.246	0.1149	0.1567
	2	0.2105	0.154	0.1779	0.2105	0.154	0.1779
	3	0.229	0.1575	**0.1867**	0.229	0.1575	**0.1867**
OrangePlus	1	0.1356	0.1095	0.1211	0.0766	0.1837	0.1081
	2	0.1886	0.1247	0.1502	0.0961	0.1767	0.1245
	3	0.178	0.1536	0.1649	0.0934	0.2283	0.1325
PCJG	1	0.0492	0.0233	0.0307	0.0492	0.0223	0.0307
TextCC-CloudPoineer	1	0.1737	0.1247	0.1452	0.0983	0.1454	0.1173
	2	0.1696	0.1341	0.1498	0.0973	0.156	0.1198
TMU-NLP	1	0.2258	0.1032	0.1417	0.2258	0.1032	0.1417
UNIPUS-Flaubert	1	0.2848	0.1415	**0.1891**	0.2276	0.1595	0.1876
	2	0.2587	0.1372	0.1793	0.1582	0.1646	0.1613
	3	0.2014	0.1603	0.1785	0.1339	0.188	0.1564
XHJZ	1	0.1293	0.1763	0.1492	0.1293	0.1763	0.1492
	2	0.1465	0.1646	0.1550	0.1465	0.1646	0.1550
	3	0.1764	0.1646	0.1703	0.1764	0.1646	0.1703
YD_NLP	1	0.3238	0.1290	**0.1845**	0.2982	0.1372	**0.1879**
	2	0.3293	0.1263	**0.1826**	0.3132	0.1337	**0.1874**
	3	0.3386	0.1259	**0.1836**	0.3217	0.1333	**0.1885**
ZZUNLP-HAN	1	0.0027	0.0012	0.0017	0.0018	0.002	0.0019
	2	0.0009	0.0004	0.0006	0.0007	0.0008	0.0007

Table11. Results of CGED 2020 in Correction Level

6 Conclusion

This study describes the NLP-TEA 2020 shared task for Chinese grammatical error diagnosis, including task design, data preparation, performance metrics, and evaluation results. Regardless of actual performance, all submissions contribute to the common effort to develop Chinese grammatical error diagnosis system, and the individual reports in the proceedings provide useful insights into computer-assisted language learning for CFL learners.

We hope the data sets collected and annotated for this shared task can facilitate and expedite future development in this research area. Therefore, all data sets with gold standards and scoring scripts are publicly available online at http://www.cged.science.

Acknowledgments

We thank all the participants for taking part in our shared task. Lung-Hao Lee helped a lot in consultation and bidding. Xiangyu Chi, Mengyao Suo, Yuhan Wang and Shufan Zhou contributed a lot in data reviewing.

This study was supported by the projects from National Language Committee Project (YB135-90).

References

Ru-Yng Chang, Chung-Hsien Wu, and Philips Kokoh Prasetyo. 2012. Error diagnosis of Chinese sentences using inductive learning algorithm and decomposition-based testing mechanism. *ACM Transactions on Asian Language Information Processing*, 11(1), article 3.

Xiliang Cui, Bao-lin Zhang. 2011. The Principles for Building the "International Corpus of Learner Chinese". *Applied Linguistics*, 2011(2), pages 100-108.

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(ENLG'11)*, pages 1-8, Nancy, France.

Reobert Dale, Ilya Anisimoff, and George Narroway. 2012. HOO 2012: A report on the preposiiton and determiner error correction shared task. *In Proceedings of the 7th Workshop on the Innovative Use of NLP for Building Educational Applications(BEA'12)*, pages 54-62, Montreal, Canada.

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 18th Conference on Computational Natural Language Learning (CoNLL'14)*: Shared Task, pages 1-12, Baltimore, Maryland, USA.

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 17th Conference on Computational Natural Language Learning(CoNLL'13)*: Shared Task, pages 1-14, Sofia, Bulgaria.

Lung-Hao Lee, Li-Ping Chang, and Yuen-Hsien Tseng. 2016. Developing learner corpus annotation for Chinese grammatical errors. *In Proceedings of the 20th International Conference on Asian Language Processing (IALP'16)*, Tainan, Taiwan.

Lung-Hao Lee, Li-Ping Chang, Kuei-Ching Lee, Yuen-Hsien Tseng, and Hsin-Hsi Chen. 2013. Linguistic rules based Chinese error detection for second language learning. *In Proceedings of the 21st International Conference on*

Computers in Education(ICCE'13), pages 27-29, Denpasar Bali, Indonesia.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis. *In Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA'15)*, pages 1-6, Beijing, China.

Lung-Hao Lee, Liang-Chih Yu, Kuei-Ching Lee, Yuen-Hsien Tseng, Li-Ping Chang, and Hsin-Hsi Chen. 2014. A sentence judgment system for grammatical error detection. *In Proceedings of the 25th International Conference on Computational Linguistics (COLING'14)*: Demos, pages 67-70, Dublin, Ireland.

Lung-Hao Lee, Rao Gaoqi, Liang-Chih Yu, Xun, Eendong, Zhang Baolin, and Chang Li-Ping. 2016. Overview of the NLP-TEA 2016 Shared Task for Chinese Grammatical Error Diagnosis. *The Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA' 16)*, pages 1-6, Osaka, Japan.

Gaoqi Rao, Baolin Zhang, Endong Xun, Lung-Hao Lee. IJCNLP-2017 Task 1: Chinese Grammatical Error Diagnosis. *In Proceedings of the IJCNLP 2017*, Shared Tasks, Taipei, Taiwan: 1-8

Gaoqi Rao, Qi Gong, Baolin Zhang, Endong Xun. Overview of NLPTEA-2018 Share Task Chinese Grammatical Error Diagnosis. 2018. *In Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA'18), pages 42-51,* Melbourne, Australia.

Chung-Hsien Wu, Chao-Hong Liu, Matthew Harris, and Liang-Chih Yu. 2010. Sentence correction incorporating relative position and parse template language models. *IEEE Transactions on Audio, Speech, and Language Processing*, 18(6), pages 1170-1181.

Chi-Hsin Yu and Hsin-Hsi Chen. 2012. Detecting word ordering errors in Chinese sentences for learning Chinese as a foreign language. *In Proceedings of the 24th International Conference on Computational Linguistics (COLING'12)*, pages 3003-3017, Bombay, India.

Liang-Chih Yu, Lung-Hao Lee, and Li-Ping Chang. 2014. Overview of grammatical error diagnosis for learning Chinese as foreign language. *In Proceedings of the 1stWorkshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA'14)*, pages 42-47, Nara, Japan.

Bao-lin Zhang, Xiliang Cui. 2013. Design Concepts of "the Construction and Research of the Inter-language Corpus of Chinese from Global Learners". *Language Teaching and Linguistic Study*, 2013(5), pages 27-34.

Bo Zheng, Wanxiang Che, Jiang Guo, Ting Liu. 2016. Chinese Grammatical Error Diagnosis with Long Short-Term Memory Networks. *In proceedings of 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA'16)*, Osaka, Japan, December 2016, pages 49–56.

Xin Zhou, Jian Wang, Xu Xie, Changlong Sun, Luo Si. Alibaba at IJCNLP-2017 Task 2: A Boosted Deep System for Dimensional Sentiment Analysis of Chinese Phrases. *In proceedings of the IJCNLP 2017*, Shared Tasks, pages 100–110, Taipei, China.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, Ting Liu. Chinese Grammatical Error Diagnosis using Statistical and Prior Knowledge driven Features with Probabilistic Ensemble Enhancement. *In Proceedings of*

the 5th Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA'18), pages 52–59, Melbourne, Australia.

Combining ResNet and Transformer for Chinese Grammatical Error Diagnosis

†‡Shaolei Wang, †Baoxin Wang, †Jiefu Gong, ‡Zhongyuan Wang, †Xiao Hu, †Xingyi Duan, †Zizhuo Shen, †Gang Yue, †Ruiji Fu, †Dayong Wu, ‡Wanxiang Che, †Shijin Wang, †Guoping Hu, ‡Ting Liu

†Joint Laboratory of HIT and iFLYTEK Research (HFL), iFLYTEK Research, Beijing, China

†State Key Laboratory of Cognitive Intelligence, iFLYTEK Research, China

‡Research Center for Social Computing and Information Retrieval (SCIR),

Harbin Institute of Technology, Harbin, China

{slwang9, bxwang2, jfgong, xiaohu2, xyduan, zzshen, gangyue, rjfu, dywu2, sjwang3, gphu}@iflytek.com, {slwang, zywang, car, tliu}@ir.hit.edu.cn

Abstract

This paper introduces our system at NLPTEA-2020 Task: Chinese Grammatical Error Diagnosis (CGED). CGED aims to diagnose four types of grammatical errors which are missing words (M), redundant words (R), bad word selection (S) and disordered words (W). The automatic CGED system contains two parts including error detection and error correction. For error detection, our system is built on the model of multi-layer bidirectional transformer encoder and ResNet is integrated into the encoder to improve the performance. We also explore stepwise ensemble selection from libraries of models to improve the performance of the single model. For error correction, we design two models to recommend corrections for S-type and M-type errors separately. In official evaluation, our system obtains the highest F1 scores at identification level and position level for error detection, and the second-highest F1 score at correction level.

1 Introduction

Chinese language is commonly regarded as one of the most complicated languages. Compared to English, Chinese has neither singular/plural change, nor the tense changes of the verb. In addition, word segmentation usually has to be processed before deeper analysis, since word boundaries are not explicitly given in Chinese. All these problems make Chinese learning challenging to new learners. In recent years, more and more people with different language and knowledge background have become interested in learning Chinese as a second language. It is necessary to develop an automatic Chinese Grammatical Error Diagnosis (CGED) tool to help to identify and correct grammatical errors written by these people.

In order to promote the development of automatic grammatical error diagnosis in Chinese learning, the Natural Language Processing Techniques for Educational Applications (NLP-TEA) have taken CGED as one of the shared tasks since 2014. Many methods have been proposed to solve CGED task.

In this work, we introduce our system at NLPTEA-2020 CGED task. For error detection, our system is built on the model of multi-layer bidirectional transformer encoder and ResNet is integrated into the encoder to improve the performance. We also explore stepwise ensemble selection from libraries of models to improve the performance of the single model. For error correction, we design two models to recommend corrections for S-type and M-type errors separately. More specifically, we use the RoBERTa (Liu et al., 2019) and the n-gram language model for the S-type correction, and utilize a combination of pretrained masked language model and a statistical language model to generate possible correction results for M-type correction. In official evaluation, our system obtains the highest F1 scores at identification level and position level for error detection, and the second-highest F1 score at correction level.

The paper is organized as follows: Section 2 briefly introduces the CGED shared task. Section 3 talks about our methodology. Section 4 shows the experiment result. Section 5 shows the related work. Finally, the conclusion and future work are drawn in Section 6.

2 Chinese Grammatical Error Diagnosis

The goal of NLPTEA CGED task is to indicate errors in the sentences written by Chinese Foreign Language learners. The sentences contain

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 36–43

Error Type	Original Sentence	Correct Sentence
M	每个城市的超市能看到这些食品。	每个城市的超市**都**能看到这些食品。
R	我和妈妈**是**不像别的母女。	我和妈妈不像别的母女。
S	最重要的是**做**孩子想学的环境。	最重要的是**创造**孩子想学的环境。
W	“静音环境”**是对人体应该**有危害的。	“静音环境”**应该是对人体**有危害的。

Table 1: Typical Error Examples, where “M” means type of missing word, “R” means type of redundant word, “S” means type of word selection and “W” means type of disordered words.

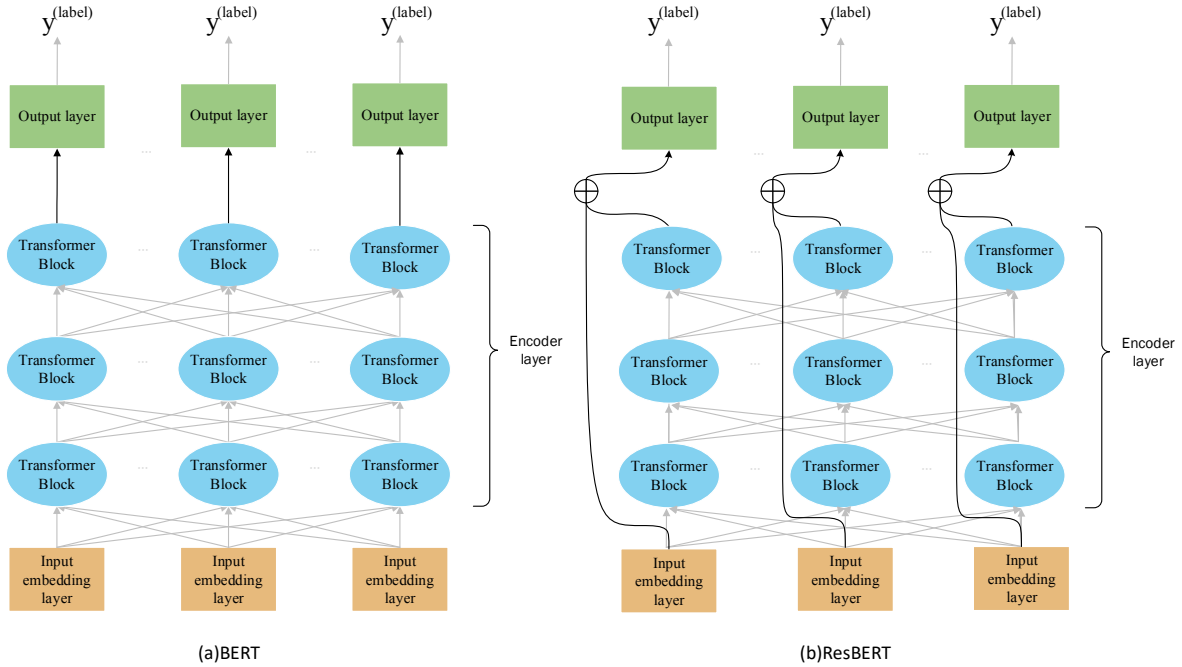


Figure 1: Architectures of BERT and ResBERT for grammatical error detection, where “BERT” means the multi-layer bidirectional transformer encoder.

four types of grammatical errors, including missing words (M), redundant words (R), word selection errors (S) and word ordering errors (W). The input sentence may contain one or more such errors. Given a sentence, the system needs to indicate: (1) If the sentence is correct or not; (2) What kind of errors the sentence contains; (3) The exact error position; (4) Possible corrections for S-type and M-type errors. Some typical examples are shown in Table 1.

3 Methodology

3.1 Error Detection

We treat the error detection problem as a sequence tagging problem. Specifically, given a sentence x, we generate a corresponding label sequence y using the BIO encoding (Kim et al., 2004). We then combine ResNet and transformer encoder to solve the tagging problem.

Transformer Encoder

We use the multi-layer bidirectional transformer encoder (BERT) described in Vaswani et al. (2017) to encode the input sentence. As shown in Figure 1(a), the model consists of three parts: an input embedding layer I, an encoder layer E and an output layer O. Given a sequence S = $w_0, ..., w_N$ as input, the encoder is formulated as follows:

$$h_i^0 = W_e w_i + W_p \quad (1)$$

$$h_i^l = \text{transformer\_block}(h_i^{l-1}) \quad (2)$$

$$y_i^{BERT} = \text{softmax}(W_o h_i^L + b_o) \quad (3)$$

where w_i is a current token, and N denotes the sequence length. Equation 1 thus creates an input embedding. Here, transformer_block includes self-attention and fully connected layers, and outputs

h_i^l. l is the number of the current layer, $l \geq 1$. L is the total number of layers of BERT. Equation 3 denotes the output layer. W_o is an output weight matrix, b_o is a bias for the output layer, and y_i^{BERT} is a grammatical error detection prediction.

Integrating ResNet

Deep neural networks learn different representations for each layer. For example, Belinkov et al. (2017) demonstrated that in a machine translation task, the low layers of the network learn to represent the word structure, while higher layers are more focused on word meaning. For tasks that emphasize the grammatical nature such as Chinese grammatical error detection, information from the lower layers is considered to be important. In this work, we use the residual learning framework (He et al., 2016) to combine the information from word embedding with the information from deep layer. Given a sequence S = $w_0, ..., w_N$ as input, ResBERT is formulated as follows:

$$h_i^0 = W_e w_i + W_p \tag{4}$$

$$h_i^l = \text{transformer\_block}(h_i^{l-1}) \tag{5}$$

$$R_i = h_i^L - w_i \tag{6}$$

$$H_i^L = \text{concat}(h_i^L, R_i) \tag{7}$$

$$y_n^{ResBERT} = \text{softmax}(W_o H_i^L + b_o) \tag{8}$$

Equation 6 denotes the residual learning framework, where the hidden output of h_i^L and the input embedding is used to approximate the residual functions. We then send the concatenation of h_i^L and R_i to the output layer.

Stepwise Ensemble Selection from Libraries of Models

We found that different random seeds and dropout values may result in different performances at the end of each training. It is straightforward to merge different model results to increase the performance. Rather than combine all the single models by weighted averaging, we use forward stepwise selection from the library of models (Caruana et al., 2004) to find a subset of models that yield excellent performance when averaged together. Library of models is generated using different random seeds and dropout values. The basic ensemble selection procedure is very simple:

1. Start with the empty ensemble.
2. Add to the ensemble the model in the library that maximizes the ensemble's performance to the Chinese grammatical error detection metric on validation set.
3. Repeat Step 2 for a fixed number of iterations or until all the models have been used.
4. Return the ensemble from the nested set of ensembles that has maximum performance on the validation set.

The voting system when selecting the best model to add at each step is span-level and it works as follow:

1. Each single model that tags a span of error text counts as a vote for that span of error text (e.g., if the word "是" in a given position, is tagged as an R-type by one single model, then it receives one vote). Note that only the spans of text that have been recognized as an error type by any of the single model are considered as candidates.
2. Each candidate span of error text is tagged as a true error if it collected a minimum number of votes, like 30% * number_of_subset_models.

The simple forward model selection procedure presented is effective, but sometimes overfits to the validation set, reducing ensemble performance on test set. To reduce the overfitting on the validation set, we make three additions to this selection procedure as described by Caruana et al. (2004):

Selection with Replacement. With model selection without replacement, performance improves as the best models are added to the ensemble, peaks, and then quickly declines. Selecting models with replacement greatly reduces this problem. Selection with replacement allows the models to be added to the ensemble multiple times. This allows selection to fine-tune ensembles by weighting models: models added to the ensemble multiple times receive more weight.

Sorted Ensemble Initialization. The simple forward model selection procedure starts with the empty ensemble. Forward selection sometimes overfits early in selection when ensembles are

small. To prevent overfitting, we sort the models in the library by their performance, and put the N best model in the ensemble before the procedure. We use N = 5.

Bagged Ensemble Selection. As the number of models in a library increases, the chances of finding combinations of models that overfit the validation set increases. Bagging can minimize this problem. We reduce the number of models by drawing a random sample of models from the library and selecting from that sample. If a particular combination of M models overfits, the probability of those M models being in a random bag of models is less than $(1-p)^M$ for p the fraction of models in the bag. We use p = 0.5, and bag ensemble selection 20 times to insure that the best models will have many opportunities to be selected. The final ensemble is the average of the 20 ensembles.

3.2 Error Correction

The systems are also required to recommend corrections for S-type and M-type errors. In this work, we design two different models to recommend corrections for S-type and M-type errors separately. We will describe them separately.

S-type Correction

For the S-type correction, we mainly use the RoBERTa (Liu et al., 2019) and the n-gram language model. Firstly, we perform domain adaptation on the language model. We use CGED training sets from previous competitions to fine-tune RoBERTa-wwm, and combine the CGED data with news corpora to train a 5-gram language model.

S-type correction includes single-character correction and multi-character correction. For the single-character correction, we consider the top 20 generated results of RoBERTa and 3,500 most frequent characters on L2 learner corpus as candidates. We score the candidates according to the prediction probability of RoBERTa and n-gram, visual similarity, and phonological similarity (Hong et al., 2019). Afterward, we select the character with the highest score as the correction result. For the multi-character correction, we also select the top 20 characters generated by RoBERTa at each position. We put these characters together to form words and reserved those in the vocabulary as candidates. In addition to the four kinds of features at the single-character correction, we also consider Levenshtein distance between the error words and candidate words.

	Error	R	M	S	W
Train	52,312	11,548	13,931	23,014	3,769
Validation	4,871	1,060	1,269	2,156	386

Table 2: Data statistics

M-type Correction

Specially, we consider the correction of M-type errors as a cloze task and utilize a combination of pretrained masked language model and a statistical language model to generate possible correction results. Given suspected missing positions, we divide the correction process of M-type errors into two steps, firstly offering possible corrections, then evaluating and picking the most reasonable ones.

When using pretrained masked language model, We first predict the number of missing characters at the suspected M-type error position through a BERT-based sequence labeling model. Then we add the same number of [MASK] symbols as predicted to the sentence before the position. Afterward, we use BERT to predict the most likely character of each [MASK] symbol, which is considered as correction candidates. When using statistical language models, we prepared a Chinese high-frequency vocabulary of L2 learners, and supplement all possible Chinese words from this vocabulary to the suspected M-type error position, generating a series of correction candidates. To evaluate the probability of each candidate, we use them to construct modified sentences and calculate the perplexity of the original sentence and all modified sentences using a statistical language model pretrained on L2 learner corpus. If the perplexity of modified sentence is significantly lower than the perplexity of the original sentence, which is controlled by a manual threshold, we consider the candidate as a predicted correction result.

4 Experiment

4.1 Dataset

Following the work of Fu et al. (2018), We trained our single models using training units that contain both the erroneous and the corrected sentences from 2016 (HSK Track), 2017 and 2018 training data sets. CGED 2016 HSK track training set consists of 10,071 training units with a total of 24,797 grammatical errors, categorized as redundant (5,538 instances), missing (6,623), word selection (10,949) and word ordering (1,687). CGED 2017 training set consists of 10,449 training units

model	FPR	Detection level			Identification			Position		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BERT	0.6333	0.6974	0.8626	0.7713	0.5406	0.5721	0.5559	0.3362	0.3178	0.3267
BERT-WWM	0.6966	0.6826	0.8854	0.7709	0.5306	0.5894	0.5585	0.3324	0.3302	0.3313
ELECTRA	0.8530	0.6519	**0.9439**	0.7712	0.5185	0.6489	0.5764	0.3288	0.372	0.3491
ResELECTRA	0.7709	0.6680	0.9167	0.7728	0.5304	0.6520	0.5849	0.3503	0.396	0.3722
WA Ensemble	0.5675	0.7216	0.8962	0.7885	0.6175	0.5799	0.5981	**0.4871**	0.3841	0.4295
S Ensemble	**0.4333**	**0.7719**	0.8667	**0.8166**	**0.6411**	**0.6562**	**0.6486**	0.4805	**0.4693**	**0.4748**

Table 3: Validation Results using single models and ensemble methods. "S Ensemble" denotes for Stepwise ensemble model.

with a total of 26,448 grammatical errors, categorized as redundant (5,852 instances), missing (7,010), word selection (11,591) and word ordering (1,995). CGED 2018 training set consists of 1,067 grammatical errors, categorized as redundant (208 instances), missing (298), word selection (87) and word ordering (474). Table 2 shows the overall data distribution in the training data.

The sentences from 2017 testing data set are used for validation. It consists of 4,871 grammatical errors, categorized as redundant (1,060 instances), missing (1,269), word selection (2,156) and word ordering (386).

4.2 Metric

The evaluation method includes four levels:

Detection level. Determine whether a sentence is correct or not. If there is an error, the sentence is incorrect. All error types will be regarded as incorrect.

Identification level. This level could be considered as a multi-class categorization problem. The correction situation should be exactly the same as the gold standard for a given type of error.

Position level. The system results should be perfectly identical with the quadruples of the gold standard.

Correction level. Characters marked as S and M need to give correct candidates. The model recommends at most 3 correction at each error.

The following metrics are measured at detection, identification, position-level.

$$\text{FalsePositiveRate} = \frac{FP}{FP + TN} \tag{9}$$

$$\text{Accuracy} = \frac{TP + TN}{TP + FP + TN + FN} \tag{10}$$

$$\text{Precision} = \frac{TP}{TP + FP} \tag{11}$$

$$\text{Recall} = \frac{TP}{TP + FN} \tag{12}$$

$$\text{F1} = \frac{2 * \text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \tag{13}$$

Since each team is allowed to submit three results, we run the stepwise ensemble selection for three times, according to the performance on detection level, identification level, position level separately.

4.3 Training Details

We try different pre-trained model parameters as the transformer's initialization such as BERT (Devlin et al., 2018), ELECTRA discriminator (Clark et al., 2020) and BERT-WWM (Cui et al., 2019). We find that the models initialized with ELECTRA discriminator always achieve better performance. So we select ELECTRA discriminator as the transformer's initialization. More concretely, we use Chinese ELECTRA-Large discriminator model[1] with 1024 hidden units, 16 heads, 24 hidden layers, 324M parameters.

For other parameters, we use streams of 128 tokens, a mini-batch of size 64, learning rate of 2e-5 and epoch of 120. We use 16 different random seeds and 5 different dropout values for each random seed to train 80 single models for the stepwise ensemble selection.

4.4 Validation Results

As shown in Table 3, we build five baseline systems including: (1) **BERT** means single model initialized with BERT (Devlin et al., 2018); (2)

[1]https://github.com/ymcui/Chinese-ELECTRA

runs	FPR	Detection level			Identification			Position		
		Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
1	**0.1010**	0.9649	0.7409	0.8382	0.7769	0.4738	0.5886	0.4970	0.2529	0.3352
2	0.2573	0.9273	0.6213	0.6736	0.7356	0.6213	**0.6736**	0.4320	0.3514	0.3876
3	0.3257	0.9101	0.8800	0.8948	0.7320	0.6011	0.6601	0.4715	0.3536	**0.4041**
Best Team	0.0163	-	-	**0.9122**	-	-	0.6736	-	-	0.4041

Table 4: Error detection performances of Submitted Runs on Official Evaluation Testing data sets. "Best Team" row records the best scores among all participant teams at each task-specific evaluating metric.

runs	Correction Top1			Correction Top3		
	Precision	Recall	F1	Precision	Recall	F1
1	0.246	0.1149	0.1567	0.246	0.1149	0.1567
2	0.2105	0.1540	0.1779	0.2105	0.1540	0.1779
3	0.2290	0.1575	0.1867	0.2290	0.1575	0.1867
Best Team	-	-	**0.1891**	-	-	**0.1885**

Table 5: Error correction performances of Submitted Runs on Official Evaluation Testing data sets. "Best Team" row records the best scores among all participant teams at each task-specific evaluating metric.

BERT-WWM means single model initialized with BERT-WWM (Cui et al., 2019); (3) **ELECTRA** means single model initialized with ELECTRA discriminator (Clark et al., 2020); (4) **ResELECTRA** means single model with ResNet unit added; (5) **WA Ensemble** means simple weighed averaging ensemble model.

Table 3 shows the overall performances of our model on the 2017 test data. The ELECTRA single model achieves much better performance than both the BERT single model and the BERT-WWM single model. We conjecture that ELECTRA discriminator is trained without masked tokens, and this makes it more suitable for CGED task which is very sensitive to surrounding words. The ResELECTRA single model achieves more than 2 point improvements on position level over the baseline ELECTRA single model, which proves the effectiveness of integrating ResNet unit. The stepwise selection ensemble model achieves almost 10 point improvements on position level over the best ResELECTRA single model. Even compared with WA ensemble model, the stepwise selection ensemble model also achieves more than 4 point improvements.

4.5 Testing Results

Table 4 shows the performances on error detection. Our system achieves the best F1 scores at the identification level and position level. Although we achieve the highest position-level F1 score of 0.4041 among all teams, there still has a wide gap for our system to solve the Chinese grammatical error diagnosis.

Table 5 shows the performances on error correction. We achieve the second-highest correction top1 score. Since we only provide zero or one candidate word, our correction top1 score is the same as our correction top3 score.

5 Related Work

The researchers used many different methods to study the English Grammatical Error Correction task and achieved good results (Ng et al., 2014). Compared with English, the research time of Chinese grammatical error diagnosis system is short, the data sets and effective methods are lacking. Chen et al. (2013) still used n-gram as the main method, and added Web resources to improve detection performance. Lin and Chu (2015) established a scoring system using n-gram, and get better correction options. In recent years, Chinese grammatical error diagnosis has been cited as a shared task of NLPTEA CGED. Many methods are proposed to solve this task (Yu et al., 2014; Lee et al., 2015, 2016). Zheng et al. (2016) proposed a BiLSTM-CRF model based on character embedding on bigram embedding. Shiue et al. (2017) combined machine learning with traditional n-gram methods, using Bi-LSTM to detect the location of errors and adding additional linguistic information, POS, n-gram. Li et al. (2017) used Bi-LSTM to generate

the probability of each characters, and used two strategies to decide whether a character is correct or not. Liao et al. (2017) used the LSTM-CRF model to detect dependencies between outputs to better detect error messages. Yang et al. (2017) added more linguistic information on LSTM-CRF model such as POS, n-gram, PMI score and dependency features. Their system achieved the best F1-scores in identification level and position level on CGED2017 task. Fu et al. (2018) added richer features on BiLSTM-CRF model such as word segmentation, Gaussian ePMI, combination of POS and PMI. They also adopted a probabilistic ensemble approach to improve system performance. Their system achieved the best F1-score in identification level and position level on CGED2018 task.

6 Conclusion and Future Work

The paper describes our system on NLPTEA-2020 CGED task, which combines ResNet and BERT for Chinese Grammatical Error Diagnosis. We also design two different ensemble strategies to maximize the model's capability. At all six evaluating levels, we have the best F1 scores in identification level and position level, the second-highest F1 score in correction top1 level, the third-highest F1 score in detection level. In the future, we are planning to build a more powerful grammatical error diagnosis system with more training data and try to improve the system's ability by using the different cross-domain corpus.

Acknowledgments

We thank the organizers of CGED 2020 for their great job. We also thank the anonymous reviewers for insightful comments and suggestions. This work was supported by the National Key R&D Program of China via grant 2018YFB1005100, and the National Natural Science Foundation of China (NSFC) via grant 61976072, 61632011 and 61772153.

References

Yonatan Belinkov, Nadir Durrani, Fahim Dalvi, Hassan Sajjad, and James Glass. 2017. What do neural machine translation models learn about morphology? *arXiv preprint arXiv:1704.03471*.

Rich Caruana, Alexandru Niculescu-Mizil, Geoff Crew, and Alex Ksikes. 2004. Ensemble selection from libraries of models. In *Proceedings of the twenty-first international conference on Machine learning*, page 18.

Kuan-Yu Chen, Hung-Shin Lee, Chung-Han Lee, Hsin-Min Wang, and Hsin-Hsi Chen. 2013. A study of language modeling for Chinese spelling check. In *Proceedings of the Seventh SIGHAN Workshop on Chinese Language Processing*, pages 79–83, Nagoya, Japan. Asian Federation of Natural Language Processing.

Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

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*.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59.

Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. 2016. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778.

Yuzhong Hong, Xianguo Yu, Neng He, Nan Liu, and Junhui Liu. 2019. Faspell: A fast, adaptable, simple, powerful chinese spell checker based on daedecoder paradigm. In *Proceedings of the 5th Workshop on Noisy User-generated Text (W-NUT 2019)*, pages 160–169.

Jin-Dong Kim, Tomoko Ohta, Yoshimasa Tsuruoka, Yuka Tateisi, and Nigel Collier. 2004. Introduction to the bio-entity recognition task at jnlpba. In *Proceedings of the international joint workshop on natural language processing in biomedicine and its applications*, pages 70–75. Citeseer.

Lung Hao Lee, Gaoqi Rao, Liang Chih Yu, Endong Xun, and Li Ping Chang. 2016. Overview of the nlp-tea 2016 shared task for chinese grammatical error diagnosis. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA'16)*.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Guest editoral: Special issue on chinese as a

foreign language. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 20, Number 1, June 2015-Special Issue on Chinese as a Foreign Language*.

Xian Li, Peng Wang, Suixue Wang, Guanyu Jiang, and Tianyuan You. 2017. CVTE at IJCNLP-2017 task 1: Character checking system for Chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 78–83, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Quanlei Liao, Jin Wang, Jinnan Yang, and Xuejie Zhang. 2017. YNU-HPCC at IJCNLP-2017 task 1: Chinese grammatical error diagnosis using a bi-directional LSTM-CRF model. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 73–77, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Chuan-Jie Lin and Wei-Cheng Chu. 2015. A study on Chinese spelling check using confusion sets and?n-gram statistics. In *International Journal of Computational Linguistics & Chinese Language Processing, Volume 20, Number 1, June 2015-Special Issue on Chinese as a Foreign Language*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

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.

Yow-Ting Shiue, Hen-Hsen Huang, and Hsin-Hsi Chen. 2017. Detection of Chinese word usage errors for non-native Chinese learners with bidirectional LSTM. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 404–410, Vancouver, Canada. 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.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Luo Si. 2017. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Liang-Chih Yu, Lung-Hao Lee, and Li-Ping Chang. 2014. Overview of grammatical error diagnosis for learning chinese as a foreign language. In *Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications*, pages 42–47.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 49–56, Osaka, Japan. The COLING 2016 Organizing Committee.

Chinese Grammatical Error Diagnosis with Graph Convolution Network and Multi-task Learning

Yikang Luo[†] [*], **Zuyi Bao**[‡], **Chen Li**[‡] and **Rui Wang**[‡]
[†] School of Software, Shanghai Jiao Tong University, Shanghai, China
[‡]Alibaba Group
[†]luoyikang@sjtu.edu.cn
[‡]{zuyi.bzy,puji.lc,masi.wr}@alibaba-inc.com

Abstract

This paper describes our participating system on the Chinese Grammatical Error Diagnosis (CGED) 2020 shared task. For the detection subtask, we propose two BERT-based approaches 1) with syntactic dependency trees enhancing the model performance and 2) under the multi-task learning framework to combine the sequence labeling and the sequence-to-sequence (seq2seq) models. For the correction subtask, we utilize the masked language model, the seq2seq model and the spelling check model to generate corrections based on the detection results. Finally, our system achieves the highest recall rate on the top-3 correction and the second best F1 score on identification level and position level.

1 Introduction

Chinese has become an influential language all over the world. More and more people choose Chinese as a second/foreign language (CSL/CFL). Their writings usually contain grammatical errors including spelling and collocation errors. For instance, a Japanese learner may write "我苹果喜欢" (I apple like) while its correct expression should be "我喜欢苹果" (I like the apple). The inconsistency of Chinese and Japanese grammatical structures will lead to different expression order. Grammatical structure in Chinese is different from other languages and affects expression.

The previous works used to do feature engineering including pretrained features and parsing features to improve performance. In this paper, we fertilize the representations from BERT with the syntactic dependency tree and propose a multi-task learning of error detection and correction. We employ three strategies based on BERT for correction based on detection results. Experiment shows that our system is effective on both detection and correction level. Our contributions are summarized as follows:

- We propose the graph-convolutional-network-based (GCN-based) approach to improve the baseline model's understanding of syntactic dependency and introduce the sequence-to-sequence (seq2seq) model to improve the performance of the original sequence labeling task.
- We combine three approaches including the masked language model, the seq2seq and the Chinese spelling check to correct the erroneous sentences based on the detection results.
- We get the highest recall rate of the top-3 correction and the second highest F1 score at the identification level and position level of the detection.

This paper is organized as follows. Section 2 describes the CGED task. Section 3 describes our system for grammatical error detection and correction. Section 4 reports the experimental results conducted by the proposed methods. Section 5 concludes this work.

```
<TEXT id="200205215525100007_2_2x1">
所以我认为安乐死绝对不要允许。
</TEXT>
<CORRECTION>
所以我认为安乐死绝对不能被允许。
</CORRECTION>
<ERROR start_off="11" end_off="12" type="S"></ERROR>
<ERROR start_off="13" end_off="13" type="M"></ERROR>
```

Figure 1: A sample of the training data.

2 Chinese Grammatical Error Diagnosis

The CGED shared task has been held since 2014. Several sets of training data have been released written by CFL learners which contain a lot of grammatical errors. For detection, the CGED defines four types of errors: (1) R (redundant word

[*]This work was done when Yikang Luo was an intern in Alibaba Group.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 44–48

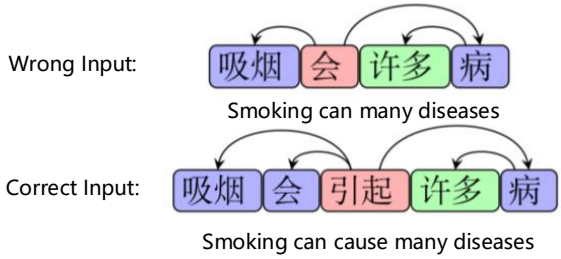


Figure 2: The different structures of syntax tree between the error sentence and right sentence.

errors);(2) M (missing words); (3) W (word ordering errors);(4) S (word selection errors) as shown in Figure 1. The performance is measured at detection level, identification level and position level. For correction, systems are required to recommend at most 3 corrections for missing and selection errors.

3 System Description

3.1 BERT-CRF

Previous works regard the detection task as the sequence labeling problem solving by the LSTM-CRF model (Huang et al., 2015). We introduce the BERT model (Devlin et al., 2018) to replace the LSTM model. For different pretrained BERT models, we choose the StructBERT (Wang et al., 2019) as our main body model. One of the reasons is that its pretraining strategy Word Structural Objective accepts sentences with wrong word order, which is similar to the word ordering errors in this task.

3.2 BERT-GCN-CRF

Previous works (Yang et al., 2017; Fu et al., 2018) spent a lot of effort in feature engineering including pretrained features and parsing features. Part-of-speech-tagging(POS), and dependency information are the most important parsing features, which indicates to us the task is closely associated with the structure of the sentence syntactic dependency. Specifically, the redundant error and the missing error sentences syntax tree are very different from the correct sentences as the Figure 2 shows.

To understand the dependency structure of an input sentence better, we introduce the Graph Convolution Network (GCN) (Kipf and Welling, 2016; Marcheggiani and Titov, 2017).

Figure 3 shows our BERT-GCN-CRF model architecture. We will explain each part in detail.

Word Dependency We split the input sentences into words and obtain the dependency relation of each word. As BERT acts on character level in Chinese, we add extra dependency edges for one word to all of characters of the word.

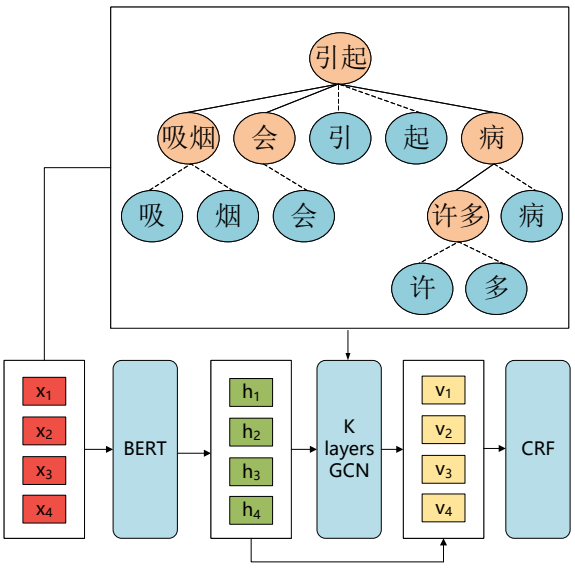


Figure 3: The structure of BERT-GCN-CRF model

Graph Convolution Network The multi-layer GCN network accepts the high-level character information obtained by the BERT model and the adjacency matrix of the dependency tree. The convolution operation is adopted for each layer.

$$f(A, H^l) = AH_l W_l^g \tag{1}$$

where $W_l^g \in R^{D\times D}$ is a trainable matrix for the l-th layer, A is the adjacency matrix of the dependency tree, $H_l = (h_1, h_2, ..., h_n)$ is the hidden state of the characters. Words use the same input representation in the network to indicate the dependency relation of the characters.

Accumulated Output After the graph convolution network, we concatenate the representation H_l for the l-th layer and the BERT hidden state passing to a linear classifier as the input of the CRF layer.

$$V = Linear(H_0 \oplus H_l) \tag{2}$$

CRF Layer A CRF layer is introduced to predict the sequence tags for each token.

$$Score(X, Y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} V_{i,y_i} \tag{3}$$

$$P(Y|X) = \frac{\exp(Score(X, Y))}{\sum_{\hat{Y}} \exp(Score(X, \hat{Y}))} \tag{4}$$

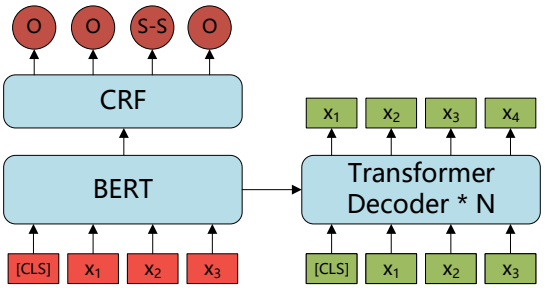


Figure 4: The structure of the multi-task learning

where X, Y, $\hat{Y}$ represents the input sequence, the truth tag sequence, and an arbitrary label sequence, V represents the emission scores, and A is the transition scores matrix of the CRF layer. The loss function is calculated as:

$$Loss_{sl} = -\log(P(Y|X)) \quad (5)$$

We use Viterbi Decoding (Huang et al., 2015) to inference answers.

3.3 Multi-task

Most previous works trained their model by the sequence tags (Yang et al., 2017; Li and Qi, 2018; Fu et al., 2018). We utilize not only tags but also correct sentences during the training process. Correct sentences are important for providing better representation in the hidden state. Moreover, with the correct sentences, the model can have a better understanding of the original meaning of the input sentence. Therefore, we introduce the seq2seq task (Sutskever et al., 2014; Vaswani et al., 2017) treating the training process as multi-task learning. As shown in Figure 4, the sequence labeling model is the encoder in our structure combined with the transformer decoders to predict the truth sentence. The sequence labeling loss and the seq2seq loss are combined by a hyper-parameter w:

$$Loss = w * Loss_{sl} + (1 - w) * Loss_{seq2seq} \quad (6)$$

During the inference phase, we use the sequence labeling module to predict answers.

3.4 Ensemble Mechanism

To take advantage of the predictions from multiple error detection models, we employ a two-stage voting ensemble mechanism.

In the first stage, predictions from multiple models are utilized to distinguish the correct sentences from the sentences with grammar errors. Specifically, we label the sentences as correct when less than θ_{det} models detect errors in the sentence.

In the second stage, an edit-level voting is applied to the predictions for the sentences with grammar errors. We only include edits that appear in the predictions of more than θ_{edit} models.

In the experiments, we use the grid search to choose the θ_{det} and θ_{edit} according to the performance on the validation data.

3.5 Correction

For the selection (S) and missing (M) errors, we introduce two methods to generate corrections.

In the first method, we insert mask tokens into the sentence and use BERT to generate correction by replacing mask tokens one by one in an auto-regressive style. In the experiments, we insert 1 to 4 mask tokens to cover most of the cases and adopt the beam-search algorithm to reduce the search complexity.

In the second method, we generate the candidates by a seq2seq model trained by mapping the wrong sentences to the correct sentences. According to the detection result, we keep generating next characters until the correct character appears within the beam-search algorithm, and then replace the incorrect span.

3.6 Chinese Spelling Check

The Chinese Spelling Check (CSC) models are utilized to handle spelling errors. We combine the results from a rule-based checker and a BERT-based spelling checker learned from the CSC data (Bao et al., To appear). The rule-based checker is good at handling non-word errors. The BERT-based checker treats the CSC task as a sequence labeling problem and is good at handling real-word errors. The corrections are then segmented and aligned with the input sentences to get the edited results on the word-level. As the CSC models show a high precision on the validation data, we treat the spelling errors as word selection errors and directly merge the CSC results into the detection and correction results for our final submissions.

4 Experiments

4.1 Data and Experiment Settings

We trained our models by CGED 2015, 2016, 2017, 2018 training data and used pairs of error sentences

Method	Detection			Identification			Position		
	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
BERT-CRF	78.4	76.7	77.5	61.4	50.8	55.6	40.7	28.7	33.6
BERT-GCN-CRF	65.5	**91.2**	76.3	53.1	**62.7**	57.5	34.7	36.1	35.4
BERT-CRF + multi-task	65.5	90.8	75.9	52.2	60.6	55.4	36.0	36.2	36.1
StrcutBERT-CRF	72.1	89.3	79.8	60.0	60.7	60.3	42.1	36.2	38.9
StrcutBERT-GCN-CRF	77.6	84.5	80.9	64.1	58.0	60.9	45.7	35.3	39.8
StrcutBERT-CRF + multi-task	73.5	88.4	80.3	60.7	62.6	61.6	42.0	38.7	40.2
Ensembled Model	**85.5**	78.6	**81.9**	**68.1**	62.1	**65.0**	**48.0**	**41.3**	**44.4**

Table 1: The results of single models and ensemble model on validation dataset.

	Detection			Identification			Position			Correction			Top-3 Correction		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Run#1	**92.8**	84.4	88.4	72.2	61.2	66.3	43.7	33.7	38.1	13.6	11.0	12.1	7.7	18.4	10.8
Run#2	91.6	**86.4**	89.0	71.9	54.5	62.0	42.4	27.3	33.2	**18.9**	12.5	15.0	**9.6**	17.7	12.5
Run#3	92.5	86.0	**89.1**	**72.3**	**62.9**	**67.3**	**44.3**	**36.1**	**39.8**	17.8	**15.3**	**16.5**	9.3	**22.8**	**13.3**
Top 1	85.7	97.6	91.2	73.6	62.1	67.4	47.2	35.4	40.4	28.5	14.2	18.9	32.2	13.3	18.9

Table 2: Final results on the official evaluation testing data. "Run #1" represents the ensemble model with correction. "Run #2" represents the single best model with correction. "Run #3" represents the ensemble model with correction and CSC. "Top 1" reports the highest F1 score with its precision and recall at different levels.

and correct sentences for the seq2seq training without extra data. We used the CGED-2018 testing dataset as our validation dataset. We introduced the BIOES (Ratinov and Roth, 2009) scheme for tagging.

Language Technology Plantform (LTP) (Che et al., 2010) was introduced to obtain the dependency tree. The hyper-parameters are selected according to the performance on the validation data through official metrics. For the GCN model, the hidden vector size was 256 with 2 layers. The batch size, learning rate, and GCN dropout were set to 32, 1e-5, 0.2. For the multi-task model, the batch size, learning rate and w are set to 32, 3e-5, 0.9.

Transformer decoder parameters are initialized from the BERT parameters as much as possible.

4.2 Validation Results

We use the BERT-CRF (base) and StructBERT-CRF (large) as our baseline models. The results of different methods are listed in Table 1. The StructBERT-CRF (large) overwhelms the BERT-CRF (base) model by obtaining a significantly better recall rate on all levels.

Both GCN and multi-task approaches achieve improved performance over the baseline model in identification level and position level. Thus, we select StrcutBERT-GCN-CRF and StructBERT-CRF + multi-task models for ensemble.

To obtain diverse single models for ensemble, we trained 38 StrcutBERT-GCN-CRF models and 65 StructBERT-CRF + multi-task models with different random seeds and hyper-parameters. As shown in Table 1, the proposed ensemble mechanism achieves an obvious improvement over the single models.

Model	Type	Precision	Recall	F1
BERT-CRF	R	**42.6**	28.3	34.0
BERT-GCN-CRF	R	36.2	**34.8**	**35.4**
BERT-CRF	M	**36.3**	26.6	30.7
BERT-GCN-CRF	M	32.8	**30.0**	**31.7**

Table 3: The position level performance of the BERT-CRF and BERT-GCN-CRF model on validation data. "R" denotes the redundant error and "M" denotes the missing error.

We evaluated the contribution of the GCN network of the redundant and missing error type. The experiment shows the effectiveness of the BERT-GCN-CRF model to resolve redundancy and missing errors.

4.3 Testing Results

For the final submission, we submitted three results from different strategies: (1) single best model with correction; (2) ensemble model with correction; (3) ensemble model with correction and CSC.

As shown in Table 2, our system approach achieves the second highest F1 scores at identification level and position level by a balanced precision and recall and highest recall rate at top-3 correction. One of the reasons for the detection gap is that for an error sentence there are multiple methods to modify the sentence and the modification granularity is difficult to control.

Most of the sentences in our training data contain grammar errors and the ensemble mechanism is tuned based on the F1 score on the validation data. These factors hurt the precision at detection level

as well as the False Positive Rate.

5 Conclusion

This article describes our system in the CGED shared task. We proposed two approaches including BERT-GCN-CRF model and multi-task learning to improve the baseline model to detect grammatical errors. We also designed three approaches including masked language model, seq2seq and spelling check to correct these errors. We got first place in the recall rate of the top-3 correction and got the second highest F1 scores at the identification level and position level.

References

Zuyi Bao, Chen Li, and Rui Wang. To appear. Chunk-based chinese spelling check with global optimization. In *Proceedings of the EMNLP 2020*.

Wanxiang Che, Zhenghua Li, and Ting Liu. 2010. Ltp: A chinese language technology platform. In *COLING 2010, 23rd International Conference on Computational Linguistics, Demonstrations Volume, 23-27 August 2010, Beijing, China*.

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

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59, Melbourne, Australia. Association for Computational Linguistics.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional lstm-crf models for sequence tagging.

Thomas N. Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks.

Changliang Li and Ji Qi. 2018. Chinese grammatical error diagnosis based on policy gradient LSTM model. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 77–82, Melbourne, Australia. Association for Computational Linguistics.

Diego Marcheggiani and Ivan Titov. 2017. Encoding sentences with graph convolutional networks for semantic role labeling. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*.

Lev Ratinov and Dan Roth. 2009. Design challenges and misconceptions in named entity recognition. In *Proceedings of the Thirteenth Conference on Computational Natural Language Learning (CoNLL-2009)*, pages 147–155, Boulder, Colorado. Association for Computational Linguistics.

Ilya Sutskever, Oriol Vinyals, and Quoc V Le. 2014. Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

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.

Wei Wang, Bin Bi, Ming Yan, Chen Wu, Zuyi Bao, Liwei Peng, and Luo Si. 2019. Structbert: Incorporating language structures into pre-training for deep language understanding.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Luo Si. 2017. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Integrating BERT and Score-based Feature Gates for Chinese Grammatical Error Diagnosis

Yongchang Cao[a] **Liang He**[a,b] **Robert Ridley**[a] **Xinyu Dai**[a]
[a]National Key Laboratory for Novel Software Technology,
Nanjing University, Nanjing, 210023, China
[b]Linguistic Intelligence and Knowledge Engineering Research, Nanjing, China
{caoyc, heliang, robertr}@smail.nju.edu.cn
daixinyu@nju.edu.cn

Abstract

This paper describes our proposed model for the Chinese Grammatical Error Diagnosis (CGED) task in NLPTEA2020. The goal of CGED is to use natural language processing techniques to automatically diagnose Chinese grammatical errors in sentences. To this end, we design and implement a CGED model named BERT with Score-feature Gates Error Diagnoser (BSGED), which is based on the BERT model, Bidirectional Long Short-Term Memory (BiLSTM) and conditional random field (CRF). In order to address the problem of losing partial-order relationships when embedding continuous feature items as with previous works, we propose a gating mechanism for integrating continuous feature items, which effectively retains the partial-order relationships between feature items. We perform LSTM processing on the encoding result of the BERT model, and further extract the sequence features. In the final test-set evaluation, we obtained the highest F1 score at the detection level and are among the top 3 F1 scores at the identification level.

1 Introduction

Recently, with the continuous development of China, more and more people have begun to learn Chinese as their second language. Due to the many complexities of Chinese, such as the differences in how tenses are formed in Chinese and English, many learners mistakenly write many Chinese sentences with grammatical errors when they first learn Chinese. Therefore, it is necessary to develop a CGED system, which can not only improve the learning efficiency of Chinese learners, but also serve many downstream tasks based on Chinese corpora.

Compared with English grammatical error diagnosis, Chinese grammatical error correction has received limited interest in the research community. English grammar error detection models began being developed as early as the 1980s, such as the early Writer's Workbench system (Macdonald NH, 1983) for detecting punctuation errors and style errors. Later, a series of tasks for English grammatical error detection and correction were proposed, such as CoNLL-2013 (Ng et al., 2013) and CoNLL-2014 (Ng et al., 2014). With the release of the CGED task in the NLPTEA workshop in recent years, grammar diagnosis models for Chinese have also begun to be developed.

The goal of the CGED task is to use natural language processing techniques to diagnose grammatical errors in Chinese sentences written by learners who use Chinese as a second language. The CGED task allows researchers to exchange experiences and ultimately promote the development of this shared task. It defines four types of Chinese grammatical errors, which are: redundant words (denoted as a capital "R"), missing words ("M"), word selection errors ("S"), and word ordering errors ("W"). The system developed for this task needs to identify the type and location of the errors in the input sentence.

Most recent solutions to the CGED shared task convert the problem into a sequence labeling problem and use a BiLSTM-CRF-based architecture as a basic framework to train the model. However, in previous work, feature engineering for the input sequence has become more and more complex. In addition, for some score-based features which exhibit partial-order relationships, such as the commonly used PMI Score features, previous works usually learn their embedding matrix after discretizing the scores. Through this process, the partial-order relationships between items will be lost, and the dimensionality of the feature embedding matrix will gradually increase as the granularity of the score

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 49–56
Suzhou, China, December 4, 2020.

discretization becomes finer, increasing the number of parameters needed to be trained. In response to the above problems, we design and implement BERT with Score-feature Gates Error Diagnoser (BSGED), and integrate score-based features through the use of a gating mechanism, which not only greatly reduces the workload of feature engineering, but also retains the original partial-order relationships for score-based features. Experiments verify that the BSGED model achieves excellent results with less feature engineering.

In summary, our contributions are as follows:

- We propose a novel model BSGED for the CGED task, which achieves better results with fewer prior features and greatly reduces the workload of feature engineering.
- We propose a gating mechanism for integrating score-based features, which not only preserves the partial-order relationships between feature items, but also greatly reduces the amount of model training parameters.
- Through ablation experiments, we verify the effectiveness of adding a BiLSTM layer to further improve the model's ability to capture long-term dependencies of input sequences.

2 Related Work

Grammatical error diagnosis models appeared as early as the 1980s. Early grammatical error diagnosis models used rule-based methods to check and correct grammatical errors (Naber D, 2003). However, because the design of matching rules requires rich linguistic knowledge, it has become more and more difficult as well as time-consuming to design rules for such models.

In order to deal with more complex error types, a series of grammatical error detection and correction models based on machine translation technology have been proposed. Brockett et al. (2006) proposed a model that uses Statistical Machine Tranalation (SMT) techniques to detect and correct grammatical errors, which deal with mass/count noun confusions by translating the incorrect phrases as a whole. Felice et al. (2014) proposed a model for grammatical error diagnosis which combines rule-based and SMT systems in a pipeline. The model first uses rules to detect errors and generate candidates. After the candidates are roughly screened by the n-gram language model, they are sent to the SMT model for further screening. In the end, candidates will be further selected through language models and filtering rules. In order to solve the CGED2018 shared task, Hu et al. (2018) proposed a sequence-to-sequence network to model the problem, and used a semi-supervised method to generate pseudo-grammatical error data for training the model.

Models based on machine translation require a large-scale training corpus to train the model. Inspired by the powerful capabilities of Neural Machine Translation (NMT) in grammatical error diagnosis, Zheng et al. (2016) regarded CGED as a sequence labeling problem, and used the powerful feature learning ability of an LSTM network to model the input sequence, and achieved better results. Yang et al. (2017) incorporated more grammatical features into the model based on the BiLSTM-CRF framework. Based on the LSTM-CRF error detection model, Li et al. (2018) combined three error correction models: a rule-based model, an NMT GEC model, and an SMT GEC model. The three GEC models aid the BiLSTM-CRF model in marking possible error locations during the detection phase. Fu et al. (2018) designed a model that incorporates richer features and added a template matcher and probability fusion mechanism.

3 Methodology

3.1 Baseline Model

Similar to most previous models for CGED shared tasks, we treat the CGED problem as a sequence labeling problem, and use BiLSTM-CRF as the basic framework of BSGED. Specifically, for a given input sequence s_i, which consists of a character sequence $[c_1, c_2, \dots, c_n]$, BSGED will output an equal-length sequence Y_i, which is composed of a label sequence $[y_1, y_2, \dots, y_n]$ composition. We adopt the BIO marking strategy, that is, for characters without grammatical errors, we mark them as 'O', and for a subsequence of grammatical errors, such as word selection errors, the initial characters will be marked as 'B-S', The remaining single characters will be marked as 'I-S'.

Inspired by previous work, we use a BiLSTM network as the RNN unit to obtain the input character encoding sequence. The BiLSTM network has a strong ability to capture long-term dependencies of the input sequence. CRFs are widely used in a large number of natural language processing tasks, especially sequence-annotation tasks. With

the addition of a CRF, the BiLSTM-CRF model can predict the input sequence more accurately. For example, the BiLSTM-CRF model can avoid incorrect sequence predictions beginning with "I-X". In terms of feature selection, we select some simple features, such as the POS tag sequence, POS Score, and PMI Score. Different from previous work, BSGED adopts the BERT model as the character encoder of the input sequence, and uses a novel fusion mechanism to incorporate score-based features. The model details are introduced in the next section. The framework of the base model adopted by BSGED is shown in Figure 1.

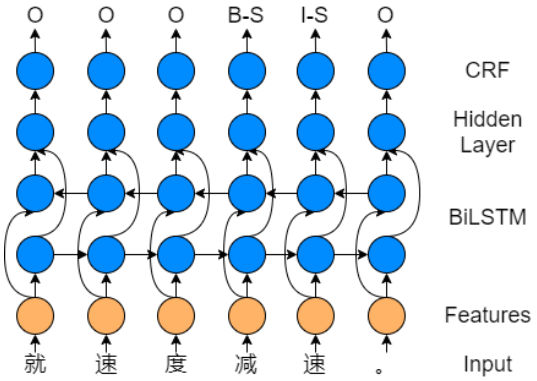


Figure 1: The base model of the BiLSTM-CRF framework used by BSGED

3.2 BERT-Encoder and Gating mechanism

Unlike previous models based on the BiLSTM-CRF architecture, BSGED does not utilize overly complex feature engineering, but uses the novel BERT model to obtain a token embedding representation of the input sequence. As a pre-trained language model, BERT has been successfully applied to many natural language understanding tasks, such as Chinese spelling error correction (Zhang et al. 2020). Due to its powerful semantic extraction capabilities, we utilize BERT as a semantic feature extractor, converting characters into vector representations. In order to preserve the long-term dependencies on the input sequence better, BSGED takes the final layer output of the BERT model as part of the BiLSTM input, instead of concatenating it with the output results of the other features through the BiLSTM network. Experiments verify that this operation can further improve the overall performance of BSGED.

We use prior knowledge to calculate the POS features of the input sequence and the PMI features between adjacent words. Specifically, we first use the LTP word segmentation tool[1] to perform word segmentation processing on the input sequence, and then perform part-of-speech tagging on the segmented sequence. This step also makes use of the LTP library. We also integrate location information into the POS tags. For example, for a Chinese sequence $A_1A_2A_3B_1B_2C_1$, the segmented sequence should be $A_1A_2A_3$ - B_1B_2 - C_1. Assuming the POS information of word A, word B, and word C are c, p, r respectively, then the result of POS labeling should be $B_cI_cI_c$ - B_pI_p - B_r.

For the score-based features, we use the news corpus provided by SogouCS[2] as a large corpus to obtain prior-knowledge statistics. Similar to the approach of Yang et al. (2017), for the POS Score feature, we first count the discrete probability distribution of the POS feature of each word, and use the probability value as its POS Score. Similarly, we count the co-occurrence frequency between every two words on the same large corpus, and use the normalized co-occurrence frequency score as the PMI Score of two adjacent words. It should be noted that we also merge the character position information in the vocabulary into these feature items.

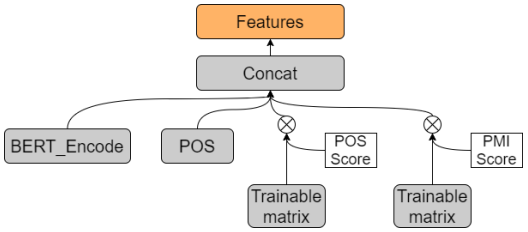


Figure 2: Schematic diagram of the features used in BSGED

We propose a novel fusion mechanism for score-based features. For continuous score features, traditional models usually discretize them first, and then embed the discretized score into a low-dimensional space. However, this embedding method will lose the partial-order relationships between the scores. In addition, the size of the feature space will change with the discretization granularity and the original value range of the score, and the model will have more parameters to be trained. Our approach differs in that we retain the continuity of

[1] https://github.com/HIT-SCIR/ltp

[2] https://www.sogou.com/labs/resource/cs.php

score features and train a matrix $\boldsymbol{M}_f \in \mathbb{R}^{2*D}$ for each score-based feature, where D is the preset embedding matrix dimension. For the i -th character, the final score embedding vector is as follows:

$$emb_i = \boldsymbol{M}_i[pos_i] * score_i \quad (1)$$

Where emb_i is the final embedding representation and pos_i is the position information of the character, $pos_i = 0$ for a "B-Word", and $pos_i = 1$ for an "I-Word". At this point, the role of score-based features is similar to an input gate (Hochreiter and Schmidhuber, 1997). This strategy not only preserves the partial-order relationship of score features, but also greatly reduces the size of the parameter matrix. The composition structure of the features for the input sequence is shown in Figure 2.

3.3 Ensemble mechanism

Following our experiments, we find that for different initialization parameters, the prediction results of the model are highly variable. This observation is consistent with that of Yang et al. (2017). In order to further improve the performance, we train multiple single models and use an ensemble mechanism to fuse them together. We adopt a simple and effective voting mechanism as our ensemble method, which improves the precision of the model while preserving the recall value.

In our final version, we use a total of four parameter groups, and we select 4 random factors for each group, so we finally merge 16 single models.

3.4 Post-Processing

The ensemble mechanism may produce conflicting prediction results. To solve this problem, we perform post-processing operations on the results of the ensemble model. We adopt some rule-base schemes, which integrate prior knowledge simply and effectively. The main processing methods are as follows:

First, in cases when some single models predict a sentence to be correct and other single models predict it to be incorrect, the conflict is resolved by retaining the prediction 'incorrect'. The 'correct' label is only output when *all* models predict the sentence as 'correct'.

Second, we resolve 'incorrect' predictions with overlaps, such as when the following two predictions are output for sentence s_i:

$$\begin{cases} < b_1, e_1, type_1 > \\ < b_2, e_2, type_2 > \end{cases} \quad (2)$$

Where b is the starting position of the prediction, e is the ending position, and $type$ is the predicted error type. When Equation 3 is established, BSGED believes that the two prediction results overlap.

$$\begin{cases} type_1 = type_2 \\ b_1 \in (b_2,\ e_2) \vee b_2 \in (b_1,\ e_1) \end{cases} \quad (3)$$

When overlapping occurs, the model uses the segmentation boundary of the original sentence to filter. Suppose that the word segmentation boundary of sentence s_i is $D = [d_1, d_2, \ldots, d_j, \ldots d_n]$, that is, $s_i[d_{j-1}: d_j]$ represents a word of the sentence. The model will retain the prediction result of $< b_i, e_i, type_i >$ which is more suitable for D.

4 Experiment

4.1 Data Preparation

We use all the data from the CGED2015-CGED2018 training and test sets, as well as the training data from CGED2020. More specifically, our training data consists of the following parts: all data from the CGED2015-2016 training set and test set, all data from the CGED2017-2018 training set, 50% of the CGED2017-2018 test set, and 20% of the CGED2020 training set. The validation set consists of 50% of the CGED2017-2018 test set and 80% of the CGED2020 training set.

Since the training set of CGED2020 has the same data as the test set from CGED2017-2018, in order to prevent data leakage, we de-duplicate the training set. Following de-duplication, the training set contains 43925 samples, and the validation set contains 3843 samples.

4.2 BERT Selection

Since richer model initialization parameters result in more diverse predictions, thereby further improving the recall rate of the model after ensembling, we therefore choose two different BERT pre-training parameters to initialize our model.

In addition to using the BERT-Base-Chinese version released by Google (Devlin et al., 2018), we also use another version of Chinese BERT. In order to further promote the research and development of Chinese information processing, the HFL team released the Chinese pre-training model BERT-wwm (Cui et al., 2019), which uses a Whole Word Masking technique, as well as models closely related to this technology: BERT-wwm-ext.

BERT-wwm is trained on Chinese Wikipedia (including simplified and traditional characters) and LTP is used to perform word segmentation before masking is carried out on all Chinese characters that make up the same word. Similar to other BERT-based models, it has 12 layers, 768 hidden size, and 12 self-attention heads.

Filter threshold	Detection Level			Identification Level			Position Level		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
1	0.7013	**0.9633**	0.8117	0.4683	**0.851**	0.6041	0.2091	**0.5419**	0.3017
4	0.8115	0.8254	**0.8184**	0.6347	0.608	**0.6211**	0.4255	0.3751	**0.3987**
10	**0.8881**	0.5408	0.6722	**0.7733**	0.3511	0.4829	**0.622**	0.2247	0.3301

Table 1: Performance of the BSGED on the validation set with different filtering thresholds

	Detection Level			Identification Level			Position Level		
	Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
Run #1	0.8565	**0.9757**	**0.9122**	0.5571	**0.8432**	0.6709	0.2097	**0.4648**	0.2890
Run #1	0.9303	0.8478	0.8872	0.7018	0.5779	0.6339	0.4008	0.288	0.3351
Run #1	0.9739	0.5513	0.7041	**0.7939**	0.2975	0.4328	**0.5757**	0.1519	0.2404
Best Team	0.9875	0.9757	0.9122	0.7939	0.8432	0.6736	0.5757	0.4648	0.4041

Table 2: The performance of the three submissions on the official evaluation test data set. The scores in bold represent the best scores we obtained among all the participating teams. The "Best Team" row records the best scores among all participating teams for each task-specific evaluating metric.

Filter threshold	Pre	Rec	F1
1	0.2091	**0.5419**	0.3017
2	0.307	0.4603	0.3683
3	0.3754	0.4149	0.3942
4	0.4255	0.3751	**0.3987**
5	0.4678	0.3439	0.3964
6	0.5071	0.3214	0.3934
7	0.5356	0.2932	0.3789
8	0.569	0.2692	0.3655
9	0.5988	0.2475	0.3502
10	**0.622**	0.2247	0.3301

Table 3: The influence of filtering threshold on the performance of the ensemble model

4.3 Validation Results

We select the model parameters through the validation set results, which mainly include the selection of the filtering threshold during model integration. Since BSGED uses a total of 16 single models for integration, we first simply set the max filtering threshold to 10, and explore the performance of the model after integration within this range. The performance of the model on the validation set is shown in Table 3 and Figure 3. It should be noted that when selecting parameters, we only paid attention to the performance of the model at the position level.

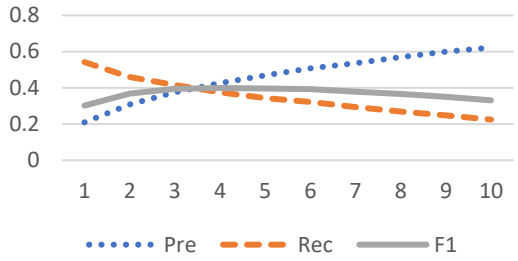


Figure 3: The influence of filtering threshold on precision, recall and F1 value.

It can be seen that as the filtering threshold increases, as does the precision, and the resulting predictions are more reliable; and as the filtering threshold decreases, the recall rate of the results will increase, enabling the model to be able to cover more actual errors. A low threshold will encourage retention of a large number of over-detection errors, while a high threshold will filter out partially correct results during post-processing. When the filter threshold is in the middle of the range, the model can achieve a higher F1 value.

Finally, we select three fusion models by selecting the parameter group with the highest precision

rate, the parameter group with the highest recall rate, and the parameter group with the highest F1 value. The results on the validation set are shown in Table 1.

4.4 Testing Results

The final version of BSGED obtained the top F1 score at the detection level and was among the top 3 F1 scores at the identification level on the test set released by CGED2020. In addition, BSGED obtained the highest precision rate and recall rate among all error diagnosis evaluation levels except the precision rate at the Detection Level. The specific results are shown in Table 2.

Single Models Num	Type	Avg. Detection Level			Avg. Identification Level			Avg. Position Level		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
7	embed	0.8416	0.694	0.7599	0.6609	0.4344	0.5238	0.4204	0.2249	0.2927
	gating	0.8264	0.7342	0.7772	0.6468	0.4958	0.5609	0.4164	0.2703	0.3275

Table 4: The influence of the gating mechanism on the model's results on the validation set. The value is the average of 7 models.

Single Models Num	Type	Avg. Detection Level			Avg. Identification Level			Avg. Position Level		
		Pre	Rec	F1	Pre	Rec	F1	Pre	Rec	F1
4	BERT -CRF	0.8298	0.7171	0.7691	0.6487	0.4575	0.5361	0.4093	0.2380	0.3006
	BSGED	0.8174	0.7556	0.7850	0.6344	0.5141	0.5677	0.4010	0.2765	0.3271

Table 5: The influence of the BiLSTM layer in the BSGED on the model's results on the validation set. The value is the average of 4 models

4.5 Ablation Experiment

In order to evaluate the novel components of our approach, we conduct two sets of ablation experiments.

The first set of ablation experiments focuses on the gating mechanism. We use 7 parameter groups from the 16 parameter groups from our original experiments. 7 single models use the traditional discretized embedding method for score-based features, and 7 single models used the novel gating approach we propose. The final comparison results are shown in Table 4. The results show that the control group that uses the gating mechanism achieves higher F1 values at each level of error detection; the performance improvements at the detection level, identification level, and position level are 0.0173, 0.0371 and 0.0348 respectively, demonstrating the effectiveness of the gating mechanism.

The second set of ablation experiments shows the performance improvement brought about by the addition of the BiLSTM layer compared to the BERT-only model. Through connecting the encoded output of the BERT model to the BiLSTM layer, the model can further improve its ability to capture the long-term dependencies of the input sequence. We conduct an experimental comparison of the model with and without the connected BiLSTM layer. For this experiment, 4 single models use a BERT-CRF architecture, and 4 single models connect the BERT output to a BiLSTM (BSGED). The two single model groups use the same parameter settings. The comparison result is shown in Table 5. As can be seen, the control group with the addition of the BiLSTM achieves F1 value improvements of 0.0159, 0.0316, and 0.0265 at the detection level, identification level, and position level, demonstrating the effectiveness of the BiLSTM layer.

4.6 Case Study

We found that different optimizations enable BSGED to solve different types of errors better. Among them, the gating mechanism directly retains the partial-order relationships of the original score-based features, so it has an improved ability for recognizing errors at character- or word-level. Some examples are shown in Table 6. For example, in the first sentence in Table 6, "多爱" (meaning

"much love") should be identified as being incorrect, with the correct phrase being "最爱" (meaning "favorite"). Similarly, "沿" (meaning "along") and "没" (meaning "no") are words formed with similar strokes. In the second example sentence, "沿" should be replaced with "没", because the PMI score of "沿有" is extremely low. In the third sentence, "速度减速" is a word-level error, and the correct expression should be "速度减慢".

The addition of the BiLSTM layer enables the model to better capture the long-term dependencies of the input sequence so that the model has stronger processing capabilities for error samples that rely on semantic understanding and long-term dependencies. Some examples are shown in Table 7. For example, in the first sentence, "在…去" should be identified as an incorrect expression in Chinese, with the correct structure being "到…去". Identifying this error that requires judging long-term dependencies of the text. Finally, the phrase "首歌" in the second example is a common collocation, but in the example, through the semantic understanding of the last clause, "首" should be identified as an R type error.

Original Sentence	**Detect Result**
我的多爱的画家也画抽象的画儿。	3, 3, S (多)
在前面，故事沿有什么特别，就是在音乐学校一个男生和一个女生交朋友。	7, 7, S (沿)
12 回合结束后，就速度减速。	12, 13, S (减速)
世界里有很多挑选新能源。最主要的生态学的能源有是：太阳能，风能，潮汐能，还有地热能。	3, 3, S (里)
首开先我们应该自问什么是成熟。对我来说，成熟就是成为负责的人，对生活的情况和问题发展自己的思考。	2, 2, R (开)

Table 6: Some examples of errors that the gating mechanism can identify but the baseline model cannot

Original Sentence	**Detect Result**
去年我们决定在挪威去。我们已经乘船去过一次挪威了。很喜欢这次航行的起点是阿姆斯特丹。	7, 7, S (在)
再说，我认为愚公当英雄，因为我们对他很尊重。香港的音乐组，他叫张华，写了一个愚公首歌。	41, 41, R (首)
星期二上午我去在大学。我学习、和我上课。下午我休息和学习在家里。星期三早上我上汉语果。	8, 8, R (在)

Table 7: Some examples of errors that the model with BiLSTM layer can identify but the baseline model cannot

5 Conclusion and Future Work

This paper describes our novel BSGED model for the CGED2020 shared task, which uses only a few and simple features, greatly reducing the workload of feature engineering for CGED; a gating mechanism is also proposed to retain the original partial-order relationships between score-based features and at the same time reduce the amount of model training parameters. In addition, we connect the sequence encoding result of the BERT model to the BiLSTM layer, which improves the BSGED model's ability to capture long-term dependencies of the input sequence. BSGED achieves the best F1 score at the detection level and the third highest F1 score at the identification level.

In the future, we intend to use the MLM model to build a model that includes grammatical error correction, and apply the natural language generation capabilities of the pre-trained language model to the task of correcting Chinese grammatical errors. In addition, we will also integrate more explicit grammatical rules, which will also greatly help the improvement of model performance.

Acknowledgments

Special thanks to the NLP-TEA workshop for sharing work, which allows us to discuss technologies and jointly promote the development of solutions.

References

Chris Brockett, William B Dolan, and Michael Gamon. 2006. Correcting ESL errors using phrasal SMT techniques. In *Proceedings of the 21stInternational Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*. Association for Computational Linguistics, 249–256 https://www.aclweb.org/anthology/P06-1032/

Cui Y, Che W, Liu T, et al. *Pre-training with whole word masking for chinese bert*[J]. arXiv preprint arXiv:1906.08101, 2019.

Devlin J, Chang M W, Lee K, et al. *Bert: Pre-training of deep bidirectional transformers for language understanding*[J]. arXiv preprint arXiv:1810.04805, 2018. http://dx.doi.org/10.18653/v1/N19-1423

Felice M, Yuan Z, Andersen Ø E, et al. *Grammatical error correction using hybrid systems and type filtering*[C]. Association for Computational Linguistics, 2014. http://dx.doi.org/10.3115/v1/W14-1702

Fu R, Pei Z, Gong J, et al. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement[C]//*Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018: 52-59. http://dx.doi.org/10.18653/v1/W18-3707

Hochreiter S, Schmidhuber J. Long short-term memory[J]. Neural computation, 1997, 9(8): 1735-1780.

Hu Q, Zhang Y, Liu F, et al. Ling@ CASS Solution to the NLP-TEA CGED Shared Task 2018[C]//*Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018: 70-76. http://dx.doi.org/10.18653/v1/W18-3709

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The conll2013 shared task on grammatical error correction. https://www.aclweb.org/anthology/W13-3601/

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. http://dx.doi.org/10.3115/v1/W14-1701

Li C, Zhou J, Bao Z, et al. A hybrid system for Chinese grammatical error diagnosis and correction [C]//*Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018: 60-69. http://dx.doi.org/10.18653/v1/W18-3708

Macdonald N H. Human factors and behavioral science: The UNIX™ Writer's Workbench software: Rationale and design[J]. Bell System Technical Journal, 1983, 62(6): 1891-1908.

Naber D. *A rule-based style and grammar checker*[J]. 2003.

Yang Y, Xie P, Tao J, et al. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task[C]//*Proceedings of the IJCNLP 2017, Shared Tasks*. 2017: 41-46.

Zhang S, Huang H, Liu J, et al. *Spelling Error Correction with Soft-Masked BERT*[J]. arXiv preprint arXiv:2005.07421, 2020. http://dx.doi.org/10.18653/v1/2020.acl-main.82

Zheng B, Che W, Guo J, et al. Chinese grammatical error diagnosis with long short-term memory networks[C]//*Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*. 2016: 49-56. https://www.aclweb.org/anthology/W16-4907/

BERT Enhanced Neural Machine Translation and Sequence Tagging Model for Chinese Grammatical Error Diagnosis

Deng Liang [1], Chen Zheng[1], Lei Guo[1], Xin Cui[1], Xiuzhang Xiong[1,2], Hengqiao Rong[1,3], and Jinpeng Dong[1]

[1]Beijing Waiyan Online Digital Technology Co.,Ltd.
[2]Minzu University of China
[3]Catholic University of Leuven
{liangdeng, zhengchen, guolei, cuixin, dongjp}@unipus.cn
hengqiao.rong@student.kuleuven.be 921883243@163.com

Abstract

This paper presents the UNIPUS-Flaubert team's hybrid system for the NLPTEA 2020 shared task of Chinese Grammatical Error Diagnosis (CGED). As a challenging NLP task, CGED has attracted increasing attention recently and has not yet fully benefited from the powerful pre-trained BERT-based models. We explore this by experimenting with three types of models. The position-tagging models and correction-tagging models are sequence tagging models fine-tuned on pre-trained BERT-based models, where the former focuses on detecting, positioning and classifying errors, and the latter aims at correcting errors. We also utilize rich representations from BERT-based models by transferring the BERT-fused models to the correction task, and further improve the performance by pre-training on a vast size of unsupervised synthetic data. To the best of our knowledge, we are the first to introduce and transfer the BERT-fused NMT model and sequence tagging model into the Chinese Grammatical Error Correction field. Our work achieved the second-highest F1 score at the detecting errors, the best F1 score at correction top1 subtask and the second-highest F1 score at correction top3 subtask.

1 Introduction

Recently, the pre-trained language models such as BERT (Devlin et al., 2019) obtain state-of-the-art results on a wide range of natural language processing (NLP) tasks, such as text classification, reading comprehension, machine translation (Zhu et al., 2020), etc. The English Grammatical Error Correction (GEC) task also benefits from the pre-trained language models. For example, in the work of Kaneko et al. (2020), they not only follow Zhu et al. (2020) to incorporate BERT into an Encoder-Decoder model for GEC, but also maximize the benefit by additionally training BERT on GEC corpora (BERT-fuse mask) or fine-tuning BERT as a GED model (BERT-fuse GED). Another route to improve the performance of GEC is using BERT as an encoder and incorporating it into a sequence tagging model (Malmi et al., 2019; Awasthi et al., 2019; Omelianchuk et al., 2020).

In the Chinese NLP community, a variety of pre-trained Chinese language models have been proposed and publicly available (Sun et al., 2019; Cui et al., 2019, 2020). Those models are proved to have a significant improvement in a variety of down-stream tasks, including reading comprehension, natural language inference, sentiment classification, etc.

In this paper, we apply the state-of-the-art English GEC models to the CGED task. Our CGED system consists of three types of models. We propose the position-tagging model, which is a sequence tagging model with a BERT encoder, to concentrate on the error localization task. The output label consists of 8 types of tags and indicates the start and end of each error for the input sentence, but it will not tell us how to correct it in the case of S (word selection) and M (missing word) errors. The correction-tagging model (Malmi et al., 2019; Awasthi et al., 2019; Omelianchuk et al., 2020) concentrates on the error correction task, and the output label contains 8772 types of tags. The tags reveal the editing operations for each Chinese character, e.g. KEEP, DELETE, APPEND, and REPLACE. The APPEND tags (3788 in total) and REPLACE tags (4982 in total) cover most Chinese characters.

The BERT-fused model (Zhu et al., 2020) is proposed for Neural Machine Translation (NMT) task and adaptively controls the interaction between representations from BERT and each layer of the Transformer (Vaswani et al., 2017) by using the attention mechanism. (Kaneko et al., 2020) transfers the BERT-fused model to the English GEC task and further advances it. Due to time limitations,

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 57–66
Suzhou, China, December 4, 2020.

we only follow the training settings in (Zhu et al., 2020). Besides, we perform unsupervised data augmentation by introducing synthetic errors on a large amount of error-free corpora, then pair synthetic and original sentences to pre-train Transformers (Grundkiewicz et al., 2019).

This paper is organized as follows: Section 2 summarizes the recent developments in the field of CGED. Section 3 introduces the dataset we used to train the models, including human-annotated data and synthetic data. Section 4 is the overview of each component of our system, including BERT-fused NMT, position-tagging model, correction-tagging model, and error annotation tool. Section 5 describes our training and ensemble process. Section 6 discusses the result of our models and Section 7 concludes the paper.

2 Related Work

Zhao et al. (2015) used a statistical machine translation method to the CGED task and examined corpus-augmentation and explored alternative translation models including syntax-based and hierarchical phrase-based models. Zheng et al. (2016), Yang et al. (2017) and Liao et al. (2017) treat the CGED task as a sequence tagging problem to detect the grammatical errors. Li and Qi (2018) applied a policy gradient LSTM model to the CGED task. Fu et al. (2018b) built a CGED system based on a BiLSTM-CRF model and combined with rule-based templates to bring in grammatical knowledge. Hu et al. (2018) employed a sequence-to-sequence model and used pseudo data to pre-training the model. Li et al. (2018) designed a system for CGED which is composed of a BiLSTM-CRF model, an NMT model, and a statistical machine translation model to detect and correct the grammatical errors. A similar system (Zhou et al., 2018) achieved a competitive result in NLPCC 2018 shared task. Fu et al. (2018a) also treated the CGED task as a translation problem and used character-based and sub-word based NMTs to correct the grammatical errors. Li et al. (2019) and Ren et al. (2018) introduced the convolutional sequence-to-sequence model into the CGED task.

3 Datasets

Training data The datasets of the NLPTEA 2014∼2018 & 2020 shared task of CGED are corpora composed of parallel sentences written by Chinese as a Foreign Language (CFL) learners and their corrections. The source sentences are selected from the essay section of the computer-based TOCFL (Test of Chinese as a Foreign Language) and written-based HSK (Pinyin of Hanyu Shuiping Kaoshi, Test of Chinese Level). Before 2016, there are only TOCFL data written in traditional Chinese. In the dataset of 2016, we have both TOCFL and HSK data. We use the `opencc`[1] package to convert the traditional Chinese to simplified Chinese for the TOCFL corpus. Since 2017, only HSK data are provided that are all written in simplified Chinese.

The grammatical errors were manually annotated by native Chinese speakers. There are four kinds of errors: R (redundant word), M (missing word), S (word selection error), and W (word ordering error). Each error type has a different proportion in the corpus and each sentence may contain several errors. For example, in the CGED 2020 training set, W/S/R/M accounted for 7%, 42%, 23%, 28% of the total errors respectively. There are 2909 manually annotated errors in 1641 sentences, and only 2 sentences are error-free.

We also collect several external datasets from NLPCC 2018 GEC [2] and other resources [3]. The NLPCC 2018 GEC data contains more than 700,000 sentences and each sentence may be correct or have one or more candidate corrections.

Synthetic data We train BERT-fused NMT models in pre-training mode and no pre-training mode. For pre-training mode, the model is pre-trained on a large amount of synthetic data (Grundkiewicz et al., 2019). The other models did not use the synthetic data.

We first split each error-free sentence into words by a Chinese word segmentation tool [4], and then randomly select several words for each sentence. The number of selected word is the product of a probability which is sampled from the normal distribution and the number of words in the sentence. For each selected word, one of the four operations including substitution, deletion, insertion, and transposition is performed with a probability of 0.5, 0.2, 0.2, 0.1, which simulates the proportions of S, M, R, W errors in the CGED data.

For substitution, the selected word is replaced by a word that has a similar meaning, pronunciation,

[1] https://github.com/BYVoid/OpenCC

[2] http://tcci.ccf.org.cn/conference/2018/taskdata.php

[3] https://github.com/shibing624/pycorrector

[4] https://github.com/fxsjy/jieba

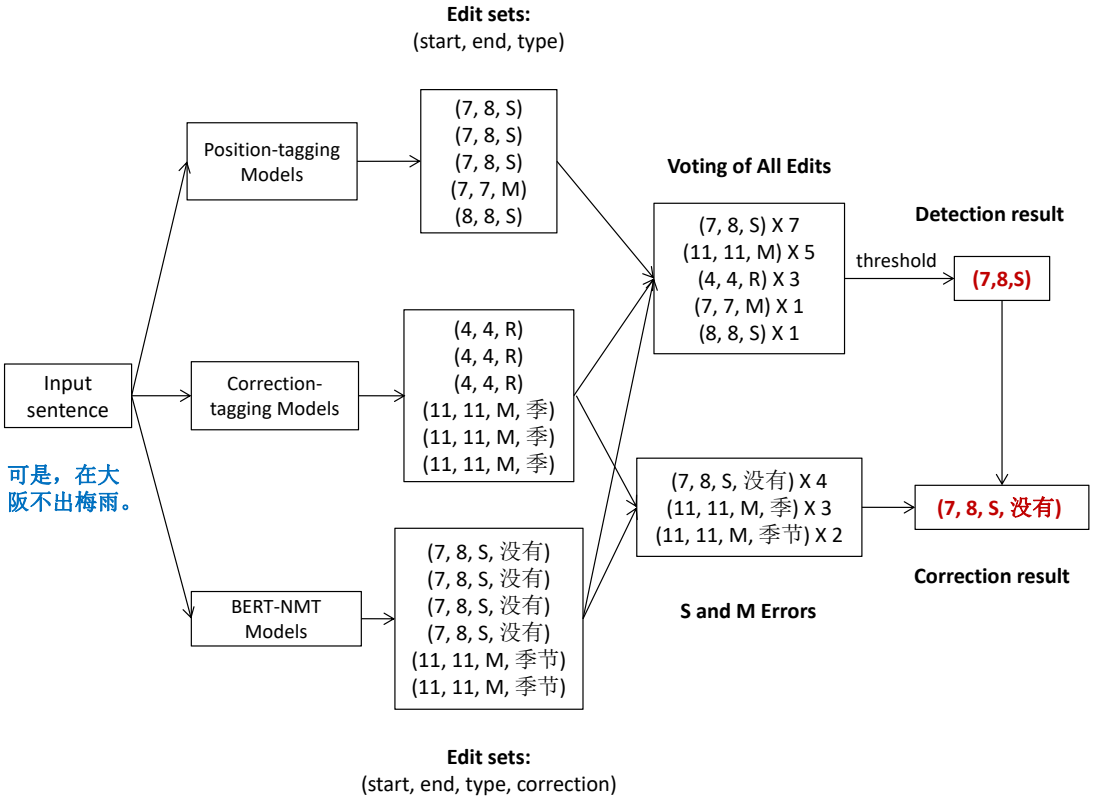


Figure 1: A demonstration of our hybrid system using a real sentence from the CGED 2020 test set. Each edit format (start, end, type, correction) stands for an error and its start position, end position, type and correction. Here, all groups of models have an equal weight 1 and the threshold is set to 7.

or shape. To simulate the confusion from similar meaning, we randomly choose a replacement from the following sources: (1) synonyms of the selected word [5] with a word similarity greater than 0.75; (2) a Chinese dictionary that we can search the word contain at least one character identical to the selected word; (3) a confusion dictionary consists of Japanese and Chinese word pairs that might be misused by Japanese learners. To mimic the confusion from similar pronunciation, we replace the selected word with a word that has the same pinyin. When introducing confusion from similar shapes, we define the similarity between two characters by their four-corner code [6].

For deletion, we simply remove the selected word. For insertion, we add on a word randomly taken from a set after the selected word. The set consists of stop words [7] and redundant words from R errors in the past CGED dataset. For transposition, we swap the selected word with the next word or with a random word in the sentence. We skip the named entities for substitution and deletion operations.

After introducing the word-level error to each error-free sentence, we introduce character-level errors by similar methods.

The corpora we used to generate synthetic data are the wiki2019zh (9.64 million sentences), the news2016zh (51.4 million sentences), the webtext2019zh (1.06 million sentences)[8] and the SogouCA (0.94 million sentences) [9].

4 System Overview

Our system consists of a sequence labeling model concentrated on the error detection subtask, and two types of error correction models aimed at generating candidate corrections.

[5] https://github.com/chatopera/Synonyms
[6] http://code.web.idv.hk/misc/four.php?i=3
[7] https://github.com/goto456/stopwords
[8] https://github.com/brightmart/nlp_chinese_corpus
[9] http://www.sogou.com/labs/resource/ca.php

Data set	# sent pairs	Source
PT	61.0 M	wiki2019zh, news2016zh
MA	1.17 M	CGED 2016∼2018 train × 5 , HSK, NLPCC
AMA	5.35 M	CGED (2014 train + test, 2015 train +test)×3, (2016 train + 2017 train)×10, 2018 train×20, HSK × 2, NLPCC × 3, webtext2019zh, SogouCA

Table 1: Summary of the three training sets we constructed to train the BERT-NMT models at different stages. The number after the multiplication sign stands for how many times the data was oversampled.

4.1 Position-tagging Model

The position-tagging model is a sequence tagging model aimed to locate grammatical errors. We use RoBERTa (Liu et al., 2019) [10] as the model's encoder then fine-tune it during training. The output tags are generated by applying a softmax layer over the encoder's logits.

Given a sequence of Chinese characters as input, the model predicts the label of each character. The output label consists of 8 types of tag, including O (correct), B-S (begin of S), I-S (middle of S), B-W (begin of W), I-W (middle of W), B-M (begin of M), B-R (begin of R), and I-R (middle of R). We extract the location and type of each error directly from the output labels. For S and M errors, the model can not give any candidate corrections.

4.2 BERT-fused NMT

The BERT-fused NMT model proposed in (Zhu et al., 2020) aims at the NMT task, we transfer the original work to the correction subtask. The BERT-fused NMT model is made up of two modules: the NMT module and the BERT module. Both modules take erroneous sentences as input. We start with training a Transformer from scratch until it converges. Then, we use the encoder and decoder of this Transformer to initialize the encoder and decoder of the NMT module. The BERT module is identical to a ready-made pre-trained BERT model.

The way to fuse the NMT module and the BERT module is to feed the representations from the BERT module (i.e. the output of the last layer of the BERT module) to each layer of the NMT module. Taking the NMT encoder as an example, the BERT-encoder attention is introduced into each NMT encoder layer and processes the representations from the BERT module. The original self-attention of each NMT encoder layer still processes the representations from the previous NMT encoder layer. The output of the BERT-encoder attention and the original self-attention are further processed by the encoder layer's original feedforward network. The NMT decoder works similarly by introducing BERT-decoder attention to each NMT decoder layer.

The parameters of the BERT-encoder attention and BERT-decoder attention are randomly initialized. During the training of the BERT-fused NMT model, the parameters of the BERT module are fixed.

4.3 Correction-tagging Model

The correction-tagging model is a sequence tagging model[11] specific to the GEC task. The output labels consist of 8772 tags, which form a large edit space. We obtain corrections by iteratively feeding a sentence to the model, getting the edit operations of each character, then editing the sentence.

To prepare the training data, we first convert the target sentence into a sequence of tags where each tag represents an edit operation on each source token. Take the following sentence pair as an example:

Source: 突 然 风 起 来 刮 了 。
Target: 突 然 刮 起 风 来 了 。

We use the minimum edit distance algorithm to align the source tokens with the target tokens. For each mapping in alignment, we collect the edit steps from the source token to the target subsequence:

突$^{\text{KEEP}}$ 然$^{\text{KEEP \& APPEND_刮 \& APPEND_起}}$
风$^{\text{KEEP}}$ 起$^{\text{DELETE}}$ 来$^{\text{KEEP}}$ 刮$^{\text{DELETE}}$
了$^{\text{KEEP}}$ 。$^{\text{KEEP}}$

Lastly, we leave only one edit for each source token, because in the training stage, each token can only have one label. In the case of the above example,

[10] RoBERTa-wwm-ext-large, from https://github.com/ymcui/Chinese-BERT-wwm

[11] https://github.com/grammarly/gector

突KEEP 然APPEND_刮 风KEEP 起DELETE
来KEEP 刮DELETE 了KEEP 。KEEP

The correction-tagging model is a pre-trained BERT-like Transformer encoder stacked with two linear layers and softmax layers on the top.

In the inference stage, we tag and edit the sentence iteratively to obtain a fully corrected sentence. In each iteration, we apply the edits according to the output labels on the input sentence and send the edited sentence to the next iteration.

4.4 Error Classification

For the BERT-fused NMT and correction-tagging model, the final output is a corrected sentence. To match with the official submission format, we align the target sentence with the source sentence to locate the start and end of the error and classify error types.

In the field of GED, there is a widely used error annotation tool — errant (Bryant et al., 2017), which automatically annotates error type information of parallel English sentences. However, there is no such tool in the CGED task. We developed a simple rule-based annotation tool to locate the error and classify the error type. Our tool first segment the source and target sentence into words using Jieba [12], then align the source and target words based on the minimum edit distance algorithm. In each mapping, if the blocks of source and target words are not the same, our tool judges this mapping as a grammatical error and determines the position and type of this error.

However, even if we have the golden corrected sentence, there exists some ambiguity when localizing and classifying the error. For example, in the CGED 2020 training set, given the following sentence pairs:

Source: 首先通过对话来知道子女的爱好、价值观，然后一起相受拥着共同的爱好。
Target: 首先通过对话来知道子女的爱好、价值观，然后一起拥有共同的爱好。

The official result is an S error starts from the 24th character and ends at the 27th character (”相受拥着”) with a correction ”拥有”. But there may be many possible solutions that depend on the word segmentation. For example, if we split ”相受拥着” into ”相受” and ”拥着”，the result becomes an R error starts from the 24th character and ends at the 25th character and an S error starts from the 26th character and end the 27th character (”拥着”) with a substitution ”拥有”. So, it is hard to locate and classify errors unambiguously due to different word segmentation rules.

We tested our annotation tool on the CGED 2020 training data set, which are shown in Table 2. Our error annotation tool loses some precision and recall at the detection, identification, and position subtasks when annotating the error information from parallel sentences.

5 Experiments

5.1 Position-tagging Model

We trained the position-tagging models with two different combinations of CGED data and used the CGED 2016 test set as the development set. For each data combination, we tried serval models with different parameter initialization and training settings. When using CGED 2016 (HSK)∼2018 & 2020 training set and 2017 test set as the training set, we get the best performance of the F1 score on detection and identification subtask on the CGED 2018 test set. When adding the TOCFL data from 2014 to 2016 to the training set, we get the best performance of the F1 score on the position subtask(see Table 3). Four position-tagging models (two models from each data combination) are used in ensemble modeling.

5.2 BERT-fused NMT

We prepared several datasets to train the BERT-fused NMT models. The first dataset is named Pre-Training data (PT data) consisting of synthetic sentences from the wiki2019zh corpus and the news2016zh corpus. The second dataset is the Manually Annotated data (MA data) which is composed of the CGED 2016∼2018 training set, HSK, and NLPCC 2018 GEC data. We filtered out the error-free sentences in HSK and NLPCC 2018 GEC dataset and oversampled the CGED data. The last dataset is the Augmented Manually-Annotated data (AMA data) consists of oversampled MA data and synthetic sentences from the text2019zh corpus and the SogouCA corpus. See details at Table 1.

We trained BERT-fused NMT models in pre-training mode and non-pre-training mode. For non-pre-training mode, we trained the BERT-fused NMT in the following steps: (1) train a baseline

[12] https://github.com/fxsjy/jieba

	M	R	S	W	Total
Detection	0.902	0.902	0.924	0.785	1
Identification	0.909	0.914	0.930	0.801	0.899
Position	0.825	0.782	0.652	0.390	0.712

Table 2: The test results of the error annotation tool. Given an original and corrected sentence pair from CGED 2020 training set, the tool extracts the position and type of each error. We compare the output of the tool with the standard result and get the F1 scores of each error type.

Model	Detection	Identification	Position
Data comb. 1	**0.780**	**0.644**	0.399
Data comb. 2	0.776	0.641	**0.428**

Table 3: The best results of the position-tagging model on the CGED 2018 test set. The data comb. 1 is the model trained on CGED 2016 (HSK)∼2018 & 2020 training set and 2017 test set, the data comb. 2 is the model trained on more data which added TOCFL 2014∼2016 data. The former gets the best performance of the F1 score on detection and identification subtask and the latter gets the best performance on the position subtask.

	2018 test set			2020 test set		
Model name	P	R	F1	P	R	F1
BERT ∗	0.213	0.193	0.203	0.185	0.134	0.155
RoBERTa ∗	0.245	0.213	0.228	0.206	0.134	0.162
ELECTRA	0.211	0.184	0.197	0.180	0.118	0.143
XLNet	0.215	0.151	0.178	0.184	0.106	0.134
Ensemble (RoBERTa + BERT) ∗	0.237	0.227	**0.232**	0.203	0.154	**0.176**
Baseline Transformer (MA)	0.263	0.0967	0.141	0.208	0.0723	0.107
→ BERT-fused (MA) ∗	0.263	0.216	**0.256**	0.223	0.118	0.154
→ Fine-tuned on AMA ∗	0.281	0.217	0.245	0.236	0.145	**0.180**
Pre-trained Transformer (PT)	0.0953	0.0324	0.0484	0.147	0.05	0.0747
→ Fine-tuned on AMA ∗	0.219	0.218	0.219	0.184	0.135	0.155
→ BERT-fused (MA)∗	0.308	0.190	**0.235**	0.224	0.124	0.159
→ BERT-fused (AMA)	0.257	0.197	0.223	0.219	0.144	**0.174**
Ensemble	-	-	-	0.222	0.192	0.206

Table 4: The results of our correction models and the ensemble on correction top1 subtask on the CGED 2018/2020 test set. The first group shows the results of the correction-tagging model with various encoders. The second / third group shows the results of the BERT-fused NMT models in non-pre-trained / pre-trained mode. The asterisk after the model name indicates that the model participates in the final ensemble. The model BERT-fused (AMA) in the third group is not used in the ensemble stage due to the time limit of the competition, and the training was completed after the deadline. The original scores of the ensemble on the CGED 2020 test set are P = 0.2848, R = 0.1415, F1 = 0.1891. We recalculated scores after an update of the error annotation tool and got a slight improvement on the final performance.

Transformer from scratch on MA data; (2) train a BERT-fused model on MA data using the baseline Transformer trained in the previous step; (3) fine-tune the previous step's BERT-fused model on AMA data. For pre-training mode, we trained the model in the following steps: (1) pre-train a Transformer from scratch on PT data; (2) fine-tune the previous step's pre-trained Transformer on AMA data; (3) train a BERT-fused model using the fine-tuned Transformer from the previous step on MA data and AMA data respectively. In all the training steps above, we combined the CGED 2018 test set and the CGED 2020 training set as the development set.

We use the `fairseq` (Ott et al., 2019) to train Transformers and the `bert-nmt` to train BERT-fused models [13]. We use *Transformer Base* architecture to train all the Transformer models and reset the learning rate scheduler and optimizer parameters when training the fine-tuned Transformer and BERT-fused model. The parameters of the fine-tuned Transformer are used to initialize the encoder and decoder of the BERT-fused model. BERT-encoder attention and BERT-decoder attention are randomly initialized. We adopt the label smoothed cross-entropy as a loss function. The overall performance of each NMT model are listed in Table 4.

5.3 Correction-tagging Model

The training of the correction-tagging model is decomposed into two stages, which are inspired by Omelianchuk et al. (2020). The first stage uses all training sets from CGED 2014∼2018 and NLPCC 2018 as the training set and the CGED 2020 training set as the development set. For NLPCC 2018 training set, we discard the sentence that is correct or has more than one correction. The second stage fine-tunes on 80% CGED 2020 training set and takes the other 20% as the development set.

The difference between our training process and Omelianchuk et al. (2020) is that we do not use synthetic data to pre-train the model. It will be investigated in future work that if a pre-training step on a large synthetic data set can improve the performance of the current model.

We fine-tune four models using the BERT (Devlin et al., 2019), RoBERTa [14], ELECTRA (Clark et al., 2020) [15], and XLNet (Yang et al., 2019) [16] encoders. The learning rate for each model on the first stage is 2e-5, 2e-5, 4e-5, and 4e-5 respectively, and all 1e-5 on the second stage. In the first stage, we freeze the encoder's weights for the first epoch and set the learning rate to 1e-3.

We adjust several hyperparameters after fine-tuning the models. The first is a threshold of the KEEP tag probability. If the KEEP tag probability is greater than the threshold, we will not change the source token. The other hyperparameters are the threshold of sentence-level minimum error probability and the number of iterations. These hyperparameters are tuned on the CGED 2018 test set to trade-off precision and recall.

A simple ensemble of RoBERTa and BERT got an additional boost of the F1 score. We use BERT, RoBERTa, and their ensemble during the ensemble modeling.

Both the BERT-fused NMT models and correction-tagging models are character-based instead of word-based for two reasons. First, the Chinese word segmentation tools are usually trained on grammatical sentences and will generate unexpected word segmentation results when applied to erroneous sentences. Second, word-based models use a larger vocabulary dictionary and more data is needed to obtain well-trained models, which conflicts with the fact that CGED is obviously a low-resource task.

5.4 Ensemble Modeling

We adopt a weighted voting strategy inspired by Li et al. (2018). The output of position-tagging models provides the position and type of each error but lack corrections for S and M errors. The output of BERT-fused NMT models and correction-tagging models are corrected sentences and are converted into the official submission format using our annotation tool in Section 4.4.

First, we omit the corrections for S and M errors temporarily and vote to determine the result of the position and type of all the errors. We accept an error proposal only if it gets the votes more than a threshold. A sentence is treated as correct if all its error proposals are not accepted. Then, we fill

[13] https://github.com/bert-nmt/bert-nmt, the pre-trained BERT from https://huggingface.co/bert-base-chinese

[14] BERT-wwm-ext and RoBERTa-wwm-ext-large, from https://github.com/ymcui/Chinese-BERT-wwm

[15] ELECTRA-large, from https://github.com/ymcui/Chinese-ELECTRA

[16] XLNet-mid, from https://github.com/ymcui/Chinese-XLNet

Submition	Detection	Identification	Position	Correction top1	Correction top3
Run 1	0.8479	0.5893	**0.3140**	**0.1891**	**0.1876**
Run 2	0.8311	0.5791	0.3057	0.1793	0.1613
Run 3	**0.8966**	**0.6463**	0.2929	0.1785	0.1564

Table 5: The overall F1 scores of our three submissions.

the corrections. For each accepted S and M error, we rank the candidate corrections from the BERT-fused NMT models and correction-tagging models according to votes. We take the first three candidates as the final corrections. A demonstration of our ensemble strategy is showed in Figure 1.

Each group of models has different weights during voting. All the thresholds and weights are tuned on the CGED 2018 test set using grid search, aiming at obtaining the best F1 score in the correction top1 subtask. The official evaluation of our three submissions are described in Table 5. Run 1 got 1st place in the correction top1 subtask and 2nd place in the correction top3 subtask. The difference between Run 1 and Run 2 is that the hyperparameter of n-best in BERT-fused NMT models is set to 1 and 8 respectively. For Run 2 (n-best is 8), each BERT-fused NMT model generates 8 candidate sentences and all take part in the voting. Run 3 tried a different ensemble modeling which mainly focused on improving recall and got the 2nd place at the detection subtask.

6 Discussion

For the BERT-fused NMT models, the BERT-fused stage improves the F1 scores for both non-pre-training and pre-training mode (See Table 4). In the non-pre-training mode, fine-tuning on AMA data further improves the performance on the CGED 2020 test set. By comparing the Baseline Transformer at the non-pre-training mode with the Fine-tuned Transformer at the pre-training mode, we find a substantial improvement of the performance on both the CGED 2018 and 2020 test sets. This proves that the CGED task can benefit from pre-training on synthetic data. However, the best results of the non-pre-training mode surpass the pre-training mode unexpectedly after the BERT-fused stage. We will investigate the reason in the future work.

(Kaneko et al., 2020) (Zhao et al., 2019) demonstrated that the GED task can help improve the performance of the GEC task. Due to time limitations, we did not try to combine the detection and correction processes in our system, which can be further improved in the future work.

In the ensemble modeling, we found that FPR (False Positive Rate) decreased as the threshold in the voting stage increased. Our submissions did not rank high in the FPR subtask, since we focused on the detection and correction rather than the FPR subtask.

Compared to the methods proposed in the NLPTEA 2018 shared task of CGED, our system greatly improves the F1 score on correction top1 and correction top3 subtask on the CGED 2018 test set. This advance mainly comes from: (1) we not only fully exploit the Transformer model for the correction subtask, but also comprehensively incorporate the power of pre-trained BERT-based models into every subtask of the CGED task; (2) the low-resource problem in the GEC task restricts the performance of NMT models (Junczys-Dowmunt et al., 2018), and we address this by utilizing the power of pre-trained BERT models and synthesizing extensive artificial data.

7 Conclusion

In this work, we present our solutions to the NLPTEA 2020 shared task of CGED. Three kinds of models are used in our system: position-tagging models, BERT-fused NMT models and correction-tagging models. Our hybrid system achieved the second-highest F1 score in the detection subtask, the highest F1 score in the correction top1 subtask and the second-highest F1 score in the correction top3 subtask, which shows that the CGED task can benefit from the recent advances of pre-trained language models.

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. In *2019 Conference on Empirical Methods in Natural Language Processing*, pages 4259–4269.

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)*, volume 1, pages 793–805.

Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Shijin Wang, and Guoping Hu. 2020. Revisiting pre-trained models for chinese natural language processing. *arXiv preprint arXiv:2004.13922*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert. *arXiv preprint arXiv:1906.08101*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 4171–4186.

Kai Fu, Jin Huang, and Yitao Duan. 2018a. Youdao's winning solution to the nlpcc-2018 task 2 challenge: A neural machine translation approach to chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 341–350.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018b. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59.

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.

Qinan Hu, Yongwei Zhang, Fang Liu, and Yueguo Gu. 2018. Ling@cass solution to the nlp-tea cged shared task 2018. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 70–76.

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)*, volume 1, pages 595–606.

Masahiro Kaneko, Masato Mita, Shun Kiyono, Jun Suzuki, and Kentaro Inui. 2020. Encoder-decoder models can benefit from pre-trained masked language models in grammatical error correction. In *ACL 2020: 58th annual meeting of the Association for Computational Linguistics*, pages 4248–4254.

Changliang Li and Ji Qi. 2018. Chinese grammatical error diagnosis based on policy gradient lstm model. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 77–82.

Chen Li, Junpei Zhou, Zuyi Bao, Hengyou Liu, Guangwei Xu, and Linlin Li. 2018. A hybrid system for chinese grammatical error diagnosis and correction. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 60–69.

Si Li, Jianbo Zhao, Guirong Shi, Yuanpeng Tan, Huifang Xu, Guang Chen, Haibo Lan, and Zhiqing Lin. 2019. Chinese grammatical error correction based on convolutional sequence to sequence model. *IEEE Access*, 7:72905–72913.

Quanlei Liao, Jin Wang, Jinnan Yang, and Xuejie Zhang. 2017. Ynu-hpcc at ijcnlp-2017 task 1: Chinese grammatical error diagnosis using a bi-directional lstm-crf model. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 73–77.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692*.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing. In *2019 Conference on Empirical Methods in Natural Language Processing*, pages 5053–5064.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem N. 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.

Myle Ott, Sergey Edunov, Alexei Baevski, Angela Fan, Sam Gross, Nathan Ng, David Grangier, and Michael Auli. 2019. fairseq: A fast, extensible toolkit for sequence modeling. In *NAACL-HLT 2019: Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 48–53.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 401–410.

Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, pages 5998–6008.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Si Luo. 2017. Alibaba at ijcnlp-2017 task 1: Embedding grammatical features into lstms for chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Ruslan Salakhutdinov, and Quoc V Le. 2019. Xlnet: Generalized autoregressive pretraining for language understanding. In *NeurIPS 2019 : Thirty-third Conference on Neural Information Processing Systems*, pages 5753–5763.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. *arXiv preprint arXiv:1903.00138*.

Yinchen Zhao, Mamoru Komachi, and Hiroshi Ishikawa. 2015. Improving chinese grammatical error correction with corpus augmentation and hierarchical phrase-based statistical machine translation. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 111–116.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *NLP-TEA@COLING*, pages 49–56.

Junpei Zhou, Chen Li, Hengyou Liu, Zuyi Bao, Guangwei Xu, and Linlin Li. 2018. Chinese grammatical error correction using statistical and neural models. In *CCF International Conference on Natural Language Processing and Chinese Computing*, pages 117–128.

Jinhua Zhu, Yingce Xia, Lijun Wu, Di He, Tao Qin, Wengang Zhou, Houqiang Li, and Tieyan Liu. 2020. Incorporating bert into neural machine translation. In *ICLR 2020 : Eighth International Conference on Learning Representations*.

A Hybrid System for NLPTEA-2020 CGED Shared Task

Meiyuan Fang[*], **Kai Fu**[*], **Jiping Wang, Yang Liu, Jin Huang, Yitao Duan**
NetEase Youdao Information Technology (Beijing) Co., LTD
{fangmeiyuan, fukai, wangjp, liuyang, huangjin, duan}
@youdao.com

Abstract

This paper introduces our system at NLPTEA2020 shared task for CGED, which is able to detect, locate, identify and correct grammatical errors in Chinese writings. The system consists of three components: GED, GEC, and post processing. GED is an ensemble of multiple BERT-based sequence labeling models for handling GED tasks. GEC performs error correction. We exploit a collection of heterogenous models, including Seq2Seq, GECToR and a candidate generation module to obtain correction candidates. Finally in the post processing stage, results from GED and GEC are fused to form the final outputs. We tune our models to lean towards optimizing precision, which we believe is more crucial in practice. As a result, among the six tracks in the shared task, our system performs well in the correction tracks: measured in F1 score, we rank first, with the highest precision, in the TOP3 correction track and third in the TOP1 correction track, also with the highest precision. Ours are among the top 4 to 6 in other tracks, except for FPR where we rank 12. And our system achieves the highest precisions among the top 10 submissions at IDENTIFICATION and POSITION tracks.

1 Introduction

With the rapid growth of online education platforms and the advance of natural language processing (NLP) techniques, recent years have seen an increased interest in automatic Grammatical Error Diagnosis (GED) and Grammatical Error Correction (GEC). Shared tasks such as CoNLL-2013, CoNLL-2014 and BEA-2019 (Ng et al., 2013, 2014; Bryant et al., 2019) were held to correct grammatical errors in essays written by learners of English as a Foreign Language (EFL). State-of-the-art GEC systems for EFL learners have achieved impressive $F_{0.5}$ scores of 66.5 on CoNLL-2014 (test) and 73.7 on BEA-2019 (test).

Despite the great success of English GEC systems, Chinese Grammatical Error Detection (CGED) and Correction (CGEC) applications yet remain relatively unexplored. Chinese, on the other hand, is quite different from western languages such as English: There are more than 3,000 commonly used Chinese characters, while English has only 26 in total; Chinese uses tones to indicate various meanings, while English uses them to express emotions; Chinese emphasizes the meaning of expressions, usually resulting in short sentences without complex structure often seen in English. Due to the large number of complex characters and flexible sentence structures, Chinese is considered one of the most difficult languages in the world to learn.

Under this circumstance, the workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA) has been organizing shared tasks for CGED (Yu et al., 2014; Lee et al., 2015, 2016; Rao et al., 2017, 2018) to help learners of Chinese as a Foreign Language (CFL) since 2014. The shared tasks provide common test conditions for researchers from both industry and academia. We believe they are very beneficial to advancing CGED technology.

This paper introduces our work on this year's CGED shared task. The task requires both error detection and correction, and we use a hybrid system to handle both. It uses as building blocks models designed for various NLP tasks, including BERT-based sequence labeling models, Seq2Seq, and GECToR. We tune our models to lean towards optimizing precision, which we believe is more crucial in practice. The performance is further improved by using synthetic data generated for individual

[*]Equal contribution.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 67–77
Suzhou, China, December 4, 2020.

tasks. Our system performs well in the correction tracks: measured in F1 score, we rank first, with the highest precision, in the TOP3 correction track and third in the TOP1 correction track, also with the highest precision. Ours are among the top 4 to 6 in other tracks, except for FPR where we rank 12. And our system achieves the highest precisions among the top 10 submissions at IDENTIFICATION and POSITION tracks.

The rest of this paper is organized as follows: A brief description of the CGED shared task is given in Section 2, followed by an overview of prior work in Section 3. Section 4 introduces our system in detail, and Section 5 demonstrates the experimental results. Finally, Section 6 concludes this paper.

2 Task Description

Generally, the CGED shared task classifies grammatical errors found in Chinese writings into four different classes, *i.e.*, redundant words (R), missing words (M), word selection errors (S), word ordering errors (W). Table 1 gives some examples of the errors, which are sampled from CGED 2020 training data. It should be noted that various error types may occur more than once in one sentence.

System performance is measured at the following levels:

- Detection-level. At this level, developed systems are required to distinguish whether a sentence contains the above-mentioned errors.
- Identification-level. At this level, developed systems need to identify the exact error types embedded in input sentences.
- Position-level. At this level, in addition to the error types, developed systems are asked to provide the positional information, indicating where the specific error occurs. For example, triples (5, 5, R) and (2, 3, W) are expected for S and W errors shown in Table 1.
- Correction-level. At this level, developed systems are required to provide up to 3 potential correction candidates for S or M errors.

3 Related Work

Grammatical Error Diagnosis. GED tasks are usually treated as a kind of sequential labeling problem. The common solution to this problem is utilizing the Long Short-Term Memory (LSTM) - Conditional Random Fields (CRF) model (Yang et al., 2017; Liao et al., 2017; Fu et al., 2018b; Zhang et al., 2018; Li et al., 2018). Performance of these approaches are usually highly dependent on the handcrafted features fed into the LSTM layer. Yang et al. (2017) extracted features including characters, character-level bi-gram, Part-of-Speech (POS), POS scores, adjacent and dependent word collocations. Later in 2018, the feature sets were further enlarged by incorporating new features like word segmentation and Gaussian exact Point-wise Mutual Information (ePMI, Fu et al., 2018b).

Grammatical Error Correction. Unlike the GED tasks, GEC tasks has been mostly treated as the machine translation problem. To the best of our knowledge, the multi-layer convolutional neural network accompanied by a large language model (Chollampatt and Ng, 2018) is considered as the first Neural Machine Translation (NMT)-like approach to handle GEC tasks in English. Then Ge et al. (2018) and Grundkiewicz and Junczys-Dowmunt (2018); Fu et al. (2018b) proposed to use recurrent neural networks, while recent work (Junczys-Dowmunt et al., 2018; Grundkiewicz et al., 2019; Lichtarge et al., 2019; Fu et al., 2018a) made use of the Transformer (Vaswani et al., 2017). Specially, GECToR (Omelianchuk et al., 2020), which considered the English GEC task as a sequential labeling problem, has obtained competitive results to previous GEC systems.

4 Methodology

The overall architecture of the developed system is depicted in Fig. 1. The proposed system can be functionally divided into three parts: GED, GEC, and post-processing. The GED framework is responsible for error diagnosis at detection, identification and position levels, while the GEC framework provides possible candidates for detected S and M errors. Finally, the post-processing module takes results from the GED and GEC frameworks and fuse them into the final form of the system outputs.

4.1 Synthetic Data Generation.

Pre-training on synthetic data is crucial for the present GEC and GED tasks since the parallel training data are still extremely scarce. It is found that the proposed basic GED models, Seq2Seq GEC models and GECToR models also benefit from synthetic data. Following previous work on English GEC tasks (Zhao et al., 2019; Grundkiewicz et al.,

Error type	Erroneous sentence	Correct sentence
R	我和妈妈是不像别的母女。 (Wǒ hé mā ma shì bù xiàng bié de mǔ nǚ.)	我和妈妈不像别的母女。 (Wǒ hé mā ma bù xiàng bié de mǔ nǚ.)
M	我同意后者主张。 (Wǒ tóng yì hòu zhě zhǔ zhāng.)	我同意后者的主张。 (Wǒ tóng yì hòu zhě de zhǔ zhāng.)
S	上周我的车刮疼啊。 (Shàng zhōu wǒ de chē guā téng a.)	上周我的车被刮了。 (Shàng zhōu wǒ de chē bèi guā le.)
W	我是还在学校上班。 (Wǒ shì hái zài xué xiào shàng bān.)	我还是在学校上班。 (Wǒ hái shì zài xué xiào shàng bān.)

Table 1: Example sentences with corresponding errors. Sequences in the bracket are the corresponding transliterations.

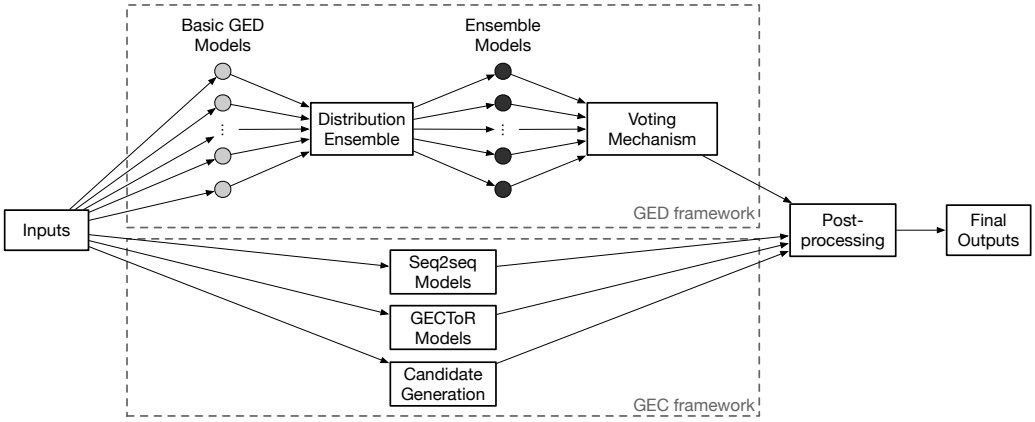


Figure 1: The overall architecture of the developed system.

2019; Xu et al., 2019), the synthetic data generation process in this work operates on two different levels, *i.e.*, word-level and character-level.

Word-level. At this level, error-free sentences are firstly segmented into words using the self-developed tokenizer. Then the following word-level errors are randomly added to the error-free sentences.

- Transposition: change the positions of words, where new positions are obtained by adding rounded bias to the original position values. The bias is sampled from a normal distribution with mean 0.0 and standard deviation 0.5.
- Deletion: delete a word.
- Insertion: add a word.
- Substitution: replace the current word with a random word in Chinese dictionary with a probability of 50%; replace the word with one of the synonyms generated by Chinese Synonyms toolkit [1] with a probability of 40%; replace the word with a word from its confusion set[2] with a probability of 10%.

The error probabilities of deletion and insertion are sampled from a normal distribution with mean 0.015 and standard deviation 0.2, while the error probability of substitution is sampled from a normal distribution with mean 0.075 and standard deviation 0.2.

Character-level. On top of the word-level errors, we also add the following character-level errors to 20% of the words, simulating spelling errors that occur in the real-world.

- Transposition: flip two consecutive characters existing in the current word with a probability of 10%.
- Deletion: delete a character in the word with a probability of 10%.
- Insertion: add a random Chinese character to the word with a probability of 10%.
- Substitution: substitute a character in the word with a probability of 30%, among which 70% of the characters are replaced by characters from their confusion sets [3], and the other 30%

[1] https://github.com/chatopera/Synonyms
[2] extracted from common mistakes made by students.
[3] http://nlp.ee.ncu.edu.tw/resource/csc.html

are replaced by random characters sampled from Chinese dictionary.

4.2 Grammatical Error Diagnosis

Basic GED Models. Recently, masked language models such as Bidirectional Encoder Representations from Transformers (BERT, Devlin et al., 2018), XLNet (Yang et al., 2019), and Generative Pre-Training 3 (GPT-3, Brown et al., 2020) have achieved superior performance on down-stream Natural Language Processing (NLP) tasks including question answering, language inference, sentence classification, *etc*.

To benefit from those efforts, we propose to use the BERT based sequential labeling model as our basic GED model rather than using the LSTM-CRF model. In general, BERT stacks 12 ($\text{BERT}_{\text{BASE}}$) or 24 ($\text{BERT}_{\text{LARGE}}$) identical Transformer blocks, which either takes a single sentence or a pair of sentences as input and outputs a hidden vector for each input token as well as a special [CLS] token for the whole input sentence (pair). Here, we denote the input sequence of Chinese characters as $\boldsymbol{X} = (x_1, x_2, ..., x_n)$, the final hidden vector generated by BERT as $\boldsymbol{H} = (h_1, h_2, ..., h_n)$, and the output BIO tags as $\boldsymbol{Y} = (y_1, y_2, ..., y_n)$. For better comprehension, we give some examples of BIO tags in Table 2. Then for an input token x_i and a specific BIO tag t, the conditional probability of x_i being labeled as t is derived using:

$$P\left(y_i = t|\boldsymbol{X}\right) = softmax\left(Wh_i + b\right). \quad (1)$$

Here, $\boldsymbol{X}$ denotes the input sequence, h_i is the final hidden state of BERT, W and b are model parameters. The tag with the largest conditional probability will be chosen as the final output corresponding to the input token x_i.

Distribution Ensemble. Top results are usually achieved by ensemble techniques (Zheng et al., 2016; Fu et al., 2018b), and this work also benefits from model ensemble approaches. Specifically, we assume that there are M different basic GED models $\{m_1, m_2, ..., m_M\}$. Then for each input sequence $\boldsymbol{X} = (x_1, x_2, ..., x_n)$, we have M output sequences $\{\boldsymbol{Y_1}, \boldsymbol{Y_2}, ..., \boldsymbol{Y_M}\}$. The distribution ensemble based on M different models can be written by:

$$P\left(y|\boldsymbol{X}\right) = \frac{1}{M}\sum_{k=1}^{M} P_k\left(y|\boldsymbol{X}; \boldsymbol{\theta_k}\right). \quad (2)$$

Here, $P\left(y|\boldsymbol{X}\right)$ denotes the conditional probability of final prediction, $\boldsymbol{\theta_k}$ indicates the trainable model parameters of kth model (m_k), and $P_k\left(y|\boldsymbol{X}; \boldsymbol{\theta_k}\right)$ is the conditional probability generated by model m_k.

Voting Mechanisms. Voting mechanisms are proposed for further improvement on overall performance, especially for model precisions. In this work, we explore the following two different voting mechanisms:

- Majority voting. In this mechanism, each output of the ensemble model is assigned the same weight, and the system selects the tag with the largest weight as the final output.

- Using F1-Score as weight. In this mechanism, we first evaluate the ensemble models using the development set and obtain corresponding F1 scores. Then the overall F1 scores serve as the weight for the ensemble models during the inference step.

4.3 Grammatical Error Correction

As shown in Figure 1, the GEC framework consists of Seq2Seq GEC models, GECToR models, and a candidates generation module.

Seq2Seq GEC Models. This work explores two kinds of Seq2Seq GEC models: one is the regular Transformer model (Vaswani et al., 2017), and the other is the copy augmented Transformer model (Zhao et al., 2019).

The attention-based Transformer is the most widely used sequence transduction model in Natural Language Processing (NLP) area that are capable of a broad spectrum of tasks (Vaswani et al., 2017; Lample et al., 2018; Yang et al., 2019; Devlin et al., 2018; Dai et al., 2019), including machine translation, text style transfer, reading comprehension, etc. Transformers employ Seq2Seq structures that are usually built up by stacking encoder and decoder layers. Encoder layers consist of a multi-head self-attention layer followed by a position-wise feed-forward layer, while decoder layers consist of a multi-head self-attention layer, a multi-head cross-attention layer and a position-wise feed-forward layer. Residual connections and layer normalizations are used to improve the performance of deep Transformers.

The copy-augmented Transformer was originally proposed for text summarization tasks (Gu et al.,

Input	因	为	,	雾	烟	刺	激	就	对	人	体	会	有	危	害	。
	(Yīn	wèi	,	wù	yān	cì	jī	jiù	duì	rén	tǐ	huì	yǒu	wēi	hài	.)
Output	O	O	O	B-S	I-S	O	O	O	B-W	I-W	I-W	I-W	O	O	O	O
Input	我	不	可	以	找	到	了	在	哪	里	我	会	买	菜	。	
	(Wǒ	bù	kě	yǐ	zhǎo	dào	le	zài	nǎ	lǐ	wǒ	huì	mǎi	cài	.)	
Output	O	B-S	I-S	I-S	O	O	B-M	B-S	O	O	B-R	I-R	O	O	O	

Table 2: Examples of BIO tags used in basic GED models. Sequences in the bracket are the corresponding transliterations.

2016; See et al., 2017) and subsequently revamped to handle GEC tasks (Zhao et al., 2019; Choe et al., 2019). Unlike the normal Transformers, copy-augmented Transformers are able to copy units (*e.g.* characters, sub-words, or words) from the source sentence, since the final probability distribution of a unit is the combination of a generative distribution and a copy distribution, balanced by a factor $\alpha^{copy} \in [0, 1]$. With a larger copy factor, the output units tend to copy from the source rather than generating their own, and vice versa.

GECToR Models. Similar to the Parallel Iterative Edit (PIE) model (Awasthi et al., 2019), GECToR (Omelianchuk et al., 2020) treats the GEC task as a sequential labeling problem. The core of the approach is the design of special output tags, which indicate the differences between source sentences and target sentences. In order to obtain the tags, minimal edits of the characters are firstly extracted based on the modified Levenshtein distance. Then the edits are converted to the following tags:

- $KEEP, indicates that the character is unchanged.
- $APPEND_X, indicates that there is a character X missing after the current character.
- $REPLACE_X, indicates that the current character should be replaced by character X.
- $REORDER, indicates that the current character is a part of the chunk where the reorder error occurs.
- $DELETE, indicates that the current character should be removed.

Identical to the basic GED models, GECToR model also stacks the fully connected layer and the softmax layer over the Transformer encoder.

Candidate Generation. During the experiment, we found that the set of correction candidates shares a large overlap across each year's training data. It is also consistent with intuition since there exist commonly confused words or characters in Chinese. To make use of this observation, we propose a candidate generation module based on a Chinese language model. Firstly, a Chinese character-level 5-gram language model (denoted by L in the following) is trained based on 30 million Chinese sentences. Then L is used to select the k most appropriate candidate words from a large set of candidates, which is extracted from the CGED training data, to replace the words in the original sentences according to the error type and position in the detection phase. Finally, the candidates along with those generated by Seq2Seq models and GECToR models are all sent to the post-processing module to obtain the final output.

4.4 Post-processing

Post-processing Outputs of GED Models. Considering that one input token is allowed to be labeled as multiple error types depending on the actual situation, we propose to apply the following heuristics to the outputs of the GED framework in the post-processing stage.

1. If current tag O is followed by a tag I-X and the last tag is B-X, where X indicates a specific error type, then the current tag will be replaced by I-X.
2. If one tag set is nested into another one, they will be decomposed based on their starting and ending points. For example, when the following case happens, (1, 4, X_1) and (2, 3, X_2) are extracted as the final outputs instead of (1, 1, X_1) and (2, 3, X_2).

Tags:	B-X_1	B-X_2	I-X_2	I-X_1
Position:	1	2	3	4

Re-ranking the Correction Candidates. To re-rank and select the elite candidates from those proposed by the three GEC models, this work proposes

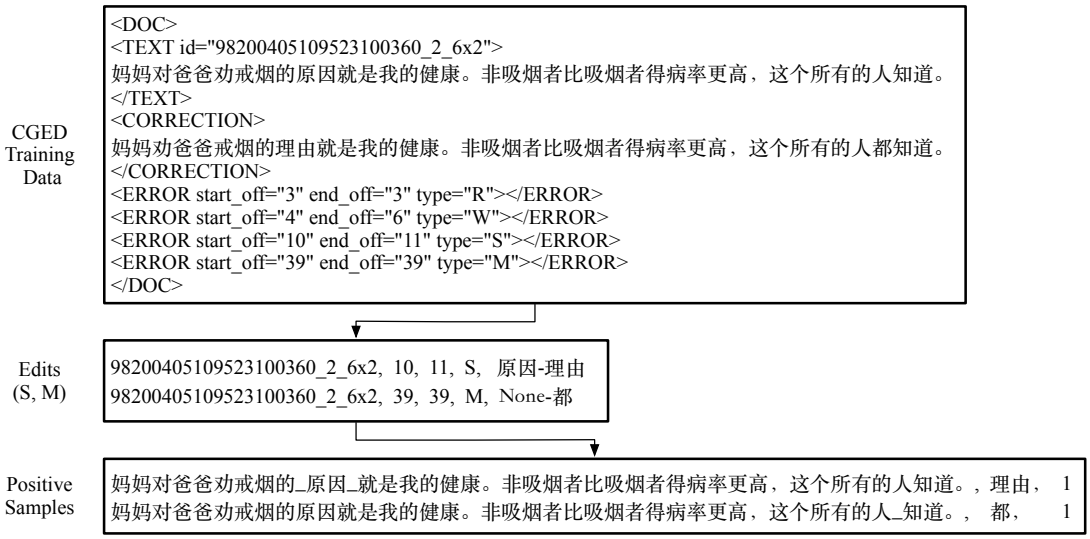


Figure 2: Example of positive data generation process.

a Chinese BERT-based [4] scoring model. The model takes a sentence and the corresponding correction candidate as input and returns the candidate's score, which lies between 0.0 and 1.0.

Since there is no ready-made data for training this kind of scoring model, it leads us to the data generation process. There are basically two kinds of data needed by the model, including positive samples and negative samples.

Positive samples can be directly generated from the CGED training data based on the process depicted in Fig. 2. We firstly design a rule-based system to extract word-level edits from the training data. Obviously, extracted edits will include all kinds of errors (R, S, W and M). However, we only keep the edits related to S and M errors, since R and W errors are not taken into consideration in the correction task. Each edit can then be converted to a training sample, which can be denoted as a triple (s, w, t), where s indicates the input sentence, w is the correction candidate, and t is the fitness score of the candidate. Specifically, for the input sentence s, we insert "_" to the left and right of the chunk where S error occurs, and we add the "_" symbol to the position where the M error occurs, as shown in Fig. 2. Considering that all training data provided by CGED shared tasks are manually annotated data, we assign higher scores (1.0 in this work) to these candidates.

The model cannot be trained only using positive samples. Hence we propose a negative data generation algorithm, as shown in Algorithm 1. Here we define D_p as the collection of all positive training data, W_p as the collection of all the candidate words in D_p, S_{pe} as the collection of all the sentences in D_p. For each input sentence s in S_{pe}, we score every word in the candidate set to find out unsuitable candidates for s. More specifically, a new sentence s_{sub}, which is reconstructed by substituting the corresponding word in s with the candidate word or inserting the candidate word to s, is scored by L. Then, we select k candidates which have the lowest scores. Finally, we randomly choose one candidate (*i.e.*, W_{cand}) from the k candidates, and form a negative sample (s, W_{cand}, 0.0) for the proposed scoring model.

Algorithm 1 Negative Training Data Generation

Input: S_{pe}, W_p, L, k
Output: negative training data D_n
$D_n \leftarrow \{\}$
$PP(t) \leftarrow$ score of sentence t calculated by L
for i in range($len\,(S_{pe})$) **do**
  $s \leftarrow S_{pe}[i]$
  for w $\in W_p$ **do**
    $s_{sub} \leftarrow$ replace the word in s with w
    $p_{s_{sub}} \leftarrow PP(s_{sub})$
  end for
  $S_{top_k} \leftarrow$ top k from W_p based on $p_{s_{sub}}$
  $w_{cand} \leftarrow random.sample(S_{top_k}, 1)$
  $D_n \leftarrow D_n + (s, w_{cand}, 0)$
end for

Inspired by the idea of "next sentence prediction" task (Devlin et al., 2018), we concatenate the input sentence s and the correction candidate w as a pair S_{pair}, and then feed it into our scoring model. Fig.3 demonstrates the architecture of the proposed scoring model as well as an example of S_{pair}.

[4] https://github.com/google-research/bert/blob/master/multilingual.md

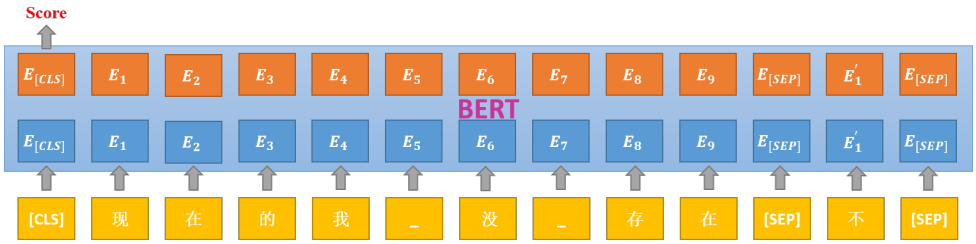


Figure 3: The architecture of the proposed scoring model.

Input	吸烟不但对自己的健康好处，而且给非吸烟者带来不好的影响。 (Xī yān bù dàn duì zì jǐ de jiàn kāng hǎo chù , ér qiě gěi fēi xī yān zhě dài lái bu hǎo de yǐng xiǎng.)
Output (Seq2Seq)	吸烟不但对自己的健康不好，而且会给非吸烟者带来不好的影响。 (Xī yān bù dàn duì zì jǐ de jiàn kāng bù hǎo, ér qiě huì gěi fēi xī yān zhě dài lái bu hǎo de yǐng xiǎng.)
Edits (Seq2Seq)	(11, 12, S, 好处-不好) (16, 16, M, None-会)
Output (GECToR)	吸烟不但对自己的健康不好，而且给不吸烟者带来不好的影响。 (Xī yān bù dàn duì zì jǐ de jiàn kāng bù hǎo, ér qiě gěi bù xī yān zhě dài lái bu hǎo de yǐng xiǎng.)
Edits (GECToR)	(11, 12, S, 好处-不好) (17, 17, M, 非-不)
Candidate Generation	(11, 12, S, 好处-不好,有害,不利)
Output (GED)	(11, 12, S)
Final Output	(11, 12, S, 好处-不好)

Table 3: Example of fusion of results. Sequences in the bracket are the corresponding transliterations.

Seq2Seq models, GECToR models and the candidate generation module tend to produce different candidates. Hence in the re-ranking stage, the correction candidate and its corresponding input sentence are fed into the scoring model one by one. We then select the top three candidates with the highest score for each input sentence.

Fusion of Results. It should be noted that the training data and vocabulary of the GEC models are different. Therefore, directly applying the ensemble techniques is infeasible. Instead, we propose to obtain the final edits by the following three steps. First, the corrected sentences produced by multiple GEC models are aligned with the original ones and the edits are extracted automatically by our rule-based extraction system. We also generate several edits with the candidate generation module based on the results of the detection phase. Second, we fuse these edits based on their error positions and types. In other words, a series of candidate words are generated for each error position. Third, we discard the edits that are not consistent with the detection results. This step is vital since the training processes of Seq2Seq models and GECToR models are completely independent and may produce conflict edits.

To improve the accuracy of correction candidates, we set a threshold to filter the candidates with less confidence. Finally, we obtain the final fusion result after all the processes described above. Table 3 shows an example of the fusion process.

Training Data	#Error	#R	#M	#S	#W
2016	48,010	9,742	14,941	20,323	3,004
2017	26,449	5,852	7,010	11,592	1,995
2018	1,067	208	298	474	87
2020	2,909	678	801	1,228	201
Total	78,435	16,480	23,050	33,617	5,287
Test Data	**#Error**	**#R**	**#M**	**#S**	**#W**
2016	7,795	1,584	2,471	3,232	508
2017	4,871	1,060	1,269	2,156	386
2018	5,040	1,119	1,381	2,167	373
2020	3,654	769	864	1,694	327

Table 4: Statistics information of the CGED data.

5 Experiments

5.1 Datasets

The proposed system utilizes training data provided by the CGED-2016, CGED-2017, CGED-2018 and CGED-2020 shared tasks. Table 4 shows the statistics of training and test data. In this work, the CGED-2016 test set is used as the training data, while CGED-2017 and CGED-2018 test sets are used as the development set. Besides the data pro-

		Detection-level			Identification-level			Position-level		
Runs	FPR	Prec.	Rec.	F1	Prec.	Rec.	F1	Prec.	Rec.	F1
1	0.2052	0.9387	0.8383	0.8857	0.7788	0.5503	0.6449	0.5145	0.2965	0.3762
2	0.2345	0.9319	0.8565	0.8926	0.7623	0.5678	0.6508	0.4822	0.3011	0.3707
3	0.2182	0.9357	0.8478	0.8896	0.7711	0.5577	0.6473	0.5011	0.2995	0.3749
Mean	0.2193	0.9354	0.8475	0.8893	0.7707	0.5586	0.6477	0.4993	0.2990	0.3739

	Correction-level (Top 1)			Correction-level (Top 3)		
Runs	Prec.	Rec.	F1	Prec.	Rec.	F1
1	0.3238	0.1290	0.1845	0.2982	0.1372	0.1879
2	0.3293	0.1263	0.1826	0.3132	0.1337	0.1874
3	0.3386	0.1259	0.1836	0.3217	0.1333	0.1885
Mean	0.3306	0.1271	0.1835	0.3110	0.1347	0.1879

Table 5: Overall performance of the developed system on CGED 2020 shared task.

vided by CGED shared task, we also utilize the data provided by the NLPCC-2018 shared task (Zhao et al., 2018)[5] to train our GEC models. Moreover, NetEase News Corpus is used to generate synthetic data.

5.2 Evaluation Metrics

As previously stated in Section 2, submitted results are evaluated at four different levels, *i.e.* detection-level, identification-level, position-level and correction-level. At each level, precision (Pre.), recall (Rec.) and F1 score are calculated based on the gold standard and the system outputs. Specially at the detection-level, false positive rate (FPR) as well as accuracy is calculated in addition to the above evaluation metrics.

5.3 Training Details

In this work, we utilize the Chinese pre-trained language models with large configuration (24 layers) including Robustly optimized BERT pre-training approach (RoBERTa, Liu et al., 2019)[6] and pre-training with whole word masking for Chinese BERT (Cui et al., 2019)[7] as the starting point of the fine-tuning process. We also tested Chinese BERT [8], but it resulted in poorer performance on the GED task than the above mentioned two models. We trained 30 basic GED models based on various pre-trained models along with different initialization seeds. Then we averaged the last several checkpoints of models and apply distribution ensemble on every 4 or 5 models. GEC models also follow similar steps to obtain final models.

5.4 Results

The overall performance of our developed system is given in Table 5. It can be seen that the system achieves F1 scores up to 0.8926, 0.6508 and 0.3762 at the detection-level, identification-level and position-level, respectively. As for the correction task, we achieve F1 scores of 0.1845 and 0.1879 at TOP1 and TOP3 correction track.

Among the 43 submissions for the detection task, our system rank 4 to 6 at detection, identification and position tracks, but rank 12 at FPR track. It is remarkable that this system achieves the highest precisions among the top 10 submissions at identification and position tracks. This system performs even better at the correction tracks. It achieves the highest F1 score also with the highest precision at TOP3 correction track. Besides, the system gets the highest precision with third-highest F1 score at TOP1 correction track, however, the gap is only 0.0046.

6 Conclusion

This paper describes our system on NLPTEA-2020 CGED shared task. To make the system more robust against data sparseness and lack of data, we adopt the synthetic data generation process during model training. Besides utilizing up-to-date model architectures, we also carefully optimized the system performance by employing ensemble techniques, voting mechanisms and rule-based post-processing. We plan to integrate more grammatical features into the GED and GEC models and optimize the post-processing algorithm to further improve the system performance.

[5] http://tcci.ccf.org.cn/conference/2018/taskdata.php

[6] https://github.com/brightmart/roberta_zh

[7] https://github.com/ymcui/Chinese-BERT-wwm

[8] https://github.com/google-research/bert/blob/master/multilingual.md

References

Abhijeet Awasthi, Sunita Sarawagi, Rasna Goyal, Sabyasachi Ghosh, and Vihari Piratla. 2019. Parallel iterative edit models for local sequence transduction. 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 4260–4270, Hong Kong, China. Association for Computational Linguistics.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners. *Computing Research Repository*, arXiv:2005.14165. Version 4.

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, Florence, Italy. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

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

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for Chinese BERT. *Computing Research Repository*, arXiv:1906.08101.

Ning Dai, Jianze Liang, Xipeng Qiu, and Xuanjing Huang. 2019. Style transformer: Unpaired text style transfer without disentangled latent representation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 5997–6007, Florence, Italy. Association for Computational Linguistics.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: Pre-training of deep bidirectional transformers for language understanding. *Computing Research Repository*, arXiv:1810.04805. Version 2.

Kai Fu, Jin Huang, and Yitao Duan. 2018a. Youdao's winning solution to the NLPCC-2018 task 2 challenge: A neural machine translation approach to Chinese grammatical error correction. In *Natural Language Processing and Chinese Computing*, pages 341–350, Cham. Springer International Publishing.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu, and Ting Liu. 2018b. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 52–59, Melbourne, Australia. Association for Computational Linguistics.

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, Melbourne, Australia. Association for Computational Linguistics.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 2 (Short Papers)*, pages 284–290, New Orleans, Louisiana. Association for Computational Linguistics.

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, Florence, Italy. Association for Computational Linguistics.

Jiatao Gu, Zhengdong Lu, Hang Li, and Victor O.K. Li. 2016. Incorporating copying mechanism in sequence-to-sequence learning. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1631–1640, Berlin, Germany. 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.

Guillaume Lample, Myle Ott, Alexis Conneau, Ludovic Denoyer, and Marc'Aurelio Ranzato. 2018. Phrase-based & neural unsupervised machine translation. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 5039–5049, Brussels, Belgium. Association for Computational Linguistics.

Lung-Hao Lee, Gaoqi Rao, Liang-Chih Yu, Endong Xun, Baolin Zhang, and Li-Ping Chang. 2016. Overview of NLP-TEA 2016 shared task for Chinese grammatical error diagnosis. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 40–48, Osaka, Japan. The COLING 2016 Organizing Committee.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis. In *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*, pages 1–6, Beijing, China. Association for Computational Linguistics.

Chen Li, Junpei Zhou, Zuyi Bao, Hengyou Liu, Guangwei Xu, and Linlin Li. 2018. A hybrid system for Chinese grammatical error diagnosis and correction. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 60–69, Melbourne, Australia. Association for Computational Linguistics.

Quanlei Liao, Jin Wang, Jinnan Yang, and Xuejie Zhang. 2017. YNU-HPCC at IJCNLP-2017 task 1: Chinese grammatical error diagnosis using a bi-directional LSTM-CRF model. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 73–77, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *Computing Research Repository*, arXiv:1907.11692.

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.

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, Seattle, WA, USA â†' Online. Association for Computational Linguistics.

Gaoqi Rao, Qi Gong, Baolin Zhang, and Endong Xun. 2018. Overview of NLPTEA-2018 share task Chinese grammatical error diagnosis. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 42–51, Melbourne, Australia. Association for Computational Linguistics.

Gaoqi Rao, Baolin Zhang, Endong Xun, and Lung-Hao Lee. 2017. IJCNLP-2017 task 1: Chinese grammatical error diagnosis. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 1–8, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Abigail See, Peter J. Liu, and Christopher D. Manning. 2017. Get to the point: Summarization with pointer-generator networks. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1073–1083, Vancouver, Canada. Association for Computational Linguistics.

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 I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems 30*, pages 5998–6008. Curran Associates, Inc.

Shuyao Xu, Jiehao Zhang, Jin Chen, and Long Qin. 2019. Erroneous data generation for grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 149–158, Florence, Italy. Association for Computational Linguistics.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Luo Si. 2017. Alibaba at IJCNLP-2017 task 1: Embedding grammatical features into LSTMs for Chinese grammatical error diagnosis task. In *Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46, Taipei, Taiwan. Asian Federation of Natural Language Processing.

Zhilin Yang, Zihang Dai, Yiming Yang, Jaime Carbonell, Russ R Salakhutdinov, and Quoc V Le. 2019. XLNet: Generalized autoregressive pretraining for

language understanding. In *Advances in Neural Information Processing Systems 32*, pages 5753–5763. Curran Associates, Inc.

Liang-Chih Yu, Lung-Hao Lee, and Liping Chang. 2014. Overview of grammatical error diagnosis for learning Chinese as a foreign language. In *Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications*, pages 42–47, Nara, Japan.

Yongwei Zhang, Qinan Hu, Fang Liu, and Yueguo Gu. 2018. CMMC-BDRC solution to the NLP-TEA-2018 Chinese grammatical error diagnosis task. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 180–187, Melbourne, Australia. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Yuanyuan Zhao, Nan Jiang, Weiwei Sun, and Xiaojun Wan. 2018. Overview of the nlpcc 2018 shared task: Grammatical error correction. In *Natural Language Processing and Chinese Computing*, pages 439–445, Cham. Springer International Publishing.

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 49–56, Osaka, Japan. The COLING 2016 Organizing Committee.

Chinese Grammatical Error Correction Based on Hybrid Models with Data Augmentation

Yi Wang[1,3,*], **Ruibin Yuan**[2,*], **YanGen Luo**[1,3,*], **YuFang Qin**[1,3,*], **NianYong Zhu**[1,3], **Peng Cheng**[1,3], and **Lihuan Wang**[1,3]

[1]State Key Laboratory of Media Convergence Production Technology and Systems Xinhua News Agency,Beijng,100077,China.

[2]Stardust.ai ,Beijing, China.

[3]The Technical Bureau of Xinhua News Agency

[*]These authors contributed to the work equally and should be regarded as co-first authors.

Abstract

A better Chinese Grammatical Error Diagnosis (CGED) system for automatic Grammatical Error Correction (GEC) can benefit foreign Chinese learners and lower Chinese learning barriers. In this paper, we introduce our solution to the CGED2020 Shared Task Grammatical Error Correction in detail. The task aims to detect and correct grammatical errors that occur in essays written by foreign Chinese learners. Our solution combined data augmentation methods, spelling check methods, and generative grammatical correction methods, and achieved the best recall score in the Top 1 Correction track. Our final result ranked fourth among the participants.

1 Introduction

In recent years, a global upsurge of Chinese learning has been set off. However, due to the language environment and language structure differences between countries, foreign Chinese learners are more prone to grammatical errors. Traditional grammatical error correction mainly relies on rule-based methods and performs poorly. Therefore, a better Chinese grammar diagnosis system is needed. Thanks to NLPTEA, the Chinese Grammatical Error Diagnosis (CGED) shared task provides a free communication platform for computing technology researchers in natural language processing (NLP) to seek more advanced Chinese grammar diagnosis solutions.

Due to the deficiency of parallel corpora, Chinese GEC often used statistical methods and rule-based methods in the early stage. Until recently, with larger-scale parallel corpora developed, machine learning techniques were applied to the Chinese GEC task. Chen(Zheng et al., 2016) used an approach based on the conditional random fields (CRF) model. The model added a collocation feature in order to better identify grammatical errors in word choice. Zhengn(Zheng et al., 2016) used a CRF based model, along with an RNN based model and an ensemble model, reached high F1-scores and recall rates across the three assessment levels of the NLP-TEA-3 shared task. Yang(Yang et al., 2017) leveraged a bi-LSTM-CRF model. Spliced word vectors with manual features such as bi-gram, POS, and PMI were added during training. Fu(Fu et al., 2018) also used a bi-LSTM-CRF model, with ePMI values integrated. They obtained promising results in the CGED2018 shared

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 78–86
Suzhou, China, December 4, 2020.

task.

In this paper, we introduce our solution to the CGED2020 shared tasks. By combining data augmentation methods, spelling check methods, and generative grammatical correction methods, we achieved the best recall score in the Top 1 Correction track. Our final result ranked fourth among the participants. The rest of this article is organized as follows: Section 2 describes the shared tasks of CGED2020. Section 3 describes the methods used in this paper, including data preprocessing, data augmentation, and various deep learning error correction models. Section 4 conducts experiments on the methods mentioned above. In Section 5, the conclusion and the planning for future works are given.

2 Task Definition

The CGED2020 shared task is the sixth Chinese grammar diagnosis error competition, held since 2014. This task provides participants a shared data set using the writing part of Hanyu Shuiping Kaoshi (HSK). The goal and direction of the task are to use modern NLP techniques to detect foreign Chinese learners' grammatical errors in Chinese writing and build an automatic Chinese grammatical error diagnosis system. It mainly distinguishes four different types of errors, including Redundant Words (R), Missing Words(M), Word Selection (S), and Word Order Error(W). On this basis, a comprehensive evaluation is carried out according to these dimensions of error judgment of the problem sentence, error type analysis, the error location, and sentence modification suggestions. The detailed sample is shown in Table 1.

The criteria for judging correctness are determined at three levels as follows.

(1) Detection-level: Binary classification of a given sentence, correct or incorrect, should be completely identical with the gold standard. All error types will be regarded as incorrect.

(2) Identification-level: This level could be considered as a multi-class categorization problem. All error types should be identified according to the gold standard.

(3) Position-level: In addition to identifying the error types, this level also judges the grammatical error's occurrence range.

3 Methodology

3.1 Data Preparation

We use the dataset from Ren (Ren et al., 2018), which contains 1.3 million sentence pairs collected from Lang-8 and HSK. Native Chinese speakers wrote news articles published by the Xinhua News Agency during 2017 and 2018, and compositions are collected for data augmentation. The former contains 6 million sentence pairs, and the latter contains 1 million sentence pairs. Texts are split into sentences, and all the non-Chinese, non-English, and non-punctuation characters in the sentences are removed. Also, sentences that are longer than 64 characters are discarded. Finally, we randomly choose 10000 pairs of sentences from non-augmented data and equally split them into validation and testing sets. The rest is used for training.

3.2 Data Augmentation

Obtaining adequate parallel data for deep learning-based GEC models is a challenging task, especially in the Chinese language. To mitigate the problem, a data augmentation scheme is applied. In this paper, we combine both rule-based and neural network-based

Error type	Error Sentence	Correct Sentence	Error location
M (missing word)	总之抽烟可以帮助所有的人了解到**对**环境的污染。	总之抽烟可以帮助所有的人了解到**它对**环境的污染。	16 - 16
R (redundant word)	现在必须得考虑怎样对待这种社会问题**的时期了**。|	现在必须得考虑怎样对待这种社会问题**了**。	18 - 20
S (word selection)	最重要的是**做**孩子想学，积极学习的环境。	最重要的是**创造**孩子想学,积极学习的环境。	6 - 6
W (word order)	**刚满 13 岁的对我**来说，流行歌曲跟我的生活非常密切。	**对刚满 13 岁的我**来说，流行歌曲跟我的生活非常密切。	1 - 7

Table 1: Error Examples

methods for generating noisy, ungrammatical texts from their clean counterparts. After the augmentation process, 6 million pairs of rule-based and 1 million pairs of neural network-based clean-corrupted sentences are obtained for training.

3.2.1 Rule Based Corruption

Inspired by previous work by Wang(Wang et al., 2019), we propose a rule-based corpora corruption method. Unlike Wang' s method, our method performs both word grain and character grain corruption and introduces sentence grain word ordering error to corpora. This method aims to obtain a large amount of parallel data with rich, diverse errors within a short time.

According to Wang, imbalanced error types will lead to low recall on the low-frequency error types. Therefore, the probability of each artificial error type are set to equal. As using low error rate data for training will cause the model to become too conservative, the corruption rate $P_{corrupt}$ is set to 0.4. With a sentence given, we obtain both character grain tokens t_c and word grain tokens t_w (using jieba). At each step, we corrupt a t_c or a t_w with a probability of $P_{corrupt}$. The corruption operations include, inserting a random character c_{rand} in the vocabulary V or a synonym syn (using synonyms) to the left of a token with a probability of p_r (redundant error type), replacing a token with a random character c_{rand} in V or a low similarity synonym syn_{low} with a probability of p_s (selection error type), deleting a token with a probability of p_m (missing error type), moving a token to a random position with a probability of p_w (word ordering error type). Algorithm 1 formalizes this method and Table 2 shows the corrupted results.

3.2.2 Neural Network Based Corruption

We train an attention-based sequence-to-sequence model with a bidirectional GRU encoder to generate noisy counterparts for clean sentences. This approach aims to generate realistic ungrammatical parallel corpora from clean corpora but is limited by the inference speed. Borrowing ideas from but being different from Xie(Xie et al., 2018) which used a noisy beam search scheme to introduce noise into the decoding stage, we define noisy score s_{noisy} as:

$$\mathrm{s_{noisy}} = \mathrm{s} - \tau\beta_{\mathrm{random}}$$

where s is the log-probability of a token, β_{random} is a scale factor and τ is a uniform random variable $\tau \sim U(0, 1)$. We use the beam size of 6 to balance between the decod-

Algorithm 1: Rule Based Corpora Corruption Method

Input: vocabulary V, clean sentences corpora C_{clean}, corruption rate $P_{corrupt}$, probability of redundant error type p_r , probability of selection error type p_s, probability of missing error type p_m, probability of word ordering error type p_w , synonym $syn()$ generator
Output: corrupted corpora C_{noisy}
Initialize $C_{noisy} = \{\}$
for each sentences s in C_{clean} ***do***
    $rand$ = Random(0,1)
    if $rand < P_{corrupt} \times p_w$ ***then***
        move a random word grain token t_w to a random position
    end
    $rand$ = Random(0,1)
    if rand $> P_{corrupt}$ ***then***
        continue
    end
    $rand$ = Random(0,1)
    if $rand < 0.5$ ***then***
        for each character grain token t_c in s ***do***
            $rand$ = Random(0,1)
            if rand $< p_r$ ***then***
            insert a token crand in V to the left of t_c
            else if $rand < p_r + p_s$ ***then***
            replace t_c with a token crand in V
            else if $rand < p_r + p_s + p_m$ ***then***
            delete t_c from s
            end if
        end
    else
        for each word grain token t_w in s ***do***
            $rand$ = Random(0,1)
            if $rand < p_r$ ***then***
            insert a synonym $syn = syn(t_w)$ to the left of t_w
            else if $rand < p_r + p_s$ ***then***
            replace t_w with a low similarity synonym $syn_{low} = syn(t_w)$
            else if $rand < p_r + p_s + p_m$ ***then***
            delete t_w from s
            end if
        end
    end if
    add s to C_{noisy}
end
return C_{noisy}

Input:	不过，特朗普无视礼仪，语出惊人，频戳痛点，不仅双边关系没能拉近，反而平添几分不和谐。
Corrupted:	不过，特朗普无视舆礼夷，语出惊人，频戳闽点，不仅双边关系没能拉近，反而平添几分炜不和谐 {。
Input:	本站比赛赛道以沙石路面为主，部分路段还设置了危险系数较高的巨石阵陡坡。
Corrupted:	本站比赛檄赛道沙石路面为主 < 部分路段还癫设叉置了步危系数较的巨石阵熘陡坡。
Input:	墨西哥总统培尼亚在首脑会议上说，今天美索美洲各国与会，正说明对话和开放是走向地区一体化的正确道路。
Corrupted:	墨西哥总统培在首脑会议上蕈, 今天鹅美洲衣与会, 渤正颢说对话妊开放是走向地区紧密结合一体化道路。
Input:	在穿着它跳舞、骑行、跑步、吃火锅时，也不会沾染汗渍或异味。
Corrupted:	在穿着跳舞、健行 =、跑步、吃火锅时“池也不会沾染狗或有毒气体]。
Input:	双方确认将敦促朝鲜遵守联合国安理会相关决议、放弃核武器和导弹计划。
Corrupted:	双方楚敦促朝鲜联合国安理会相关决议、放弃核武器和导弹计划。

Table 2: Rule based corrupted results.

ing speed and the performance. The best generation result is observed when $\beta_{random} = 3.6$. Also, coverage penalty $cp(X;Y)$ (Wu et al., 2016)is added to s_{noisy} after _EOS_ is predicted, which is computed by:

$$\mathrm{cp(X;Y)} = \beta \times \sum_{\mathrm{i=1}}^{\mathrm{|X|}} \log(\min(\sum_{\mathrm{j=1}}^{\mathrm{|Y|}} \mathrm{p_{i,j}}, 1.0))$$

where β is a scale factor, $|X|$ and $|Y|$ are the length of input sequence X and output sequence Y and $p_{i,j}$ is the attention probability of the j-th output word y_j on the i-th input word xi. Using coverage penalty can help avoid severe under-translation especially when the noisy score is used. We set $\beta = 0.3$. Examples of the neural network-based corruption results are shown in Table 3.

3.3 Generative Models

We employ two generative models, Lasertagger and Conv-Seq2Seq, for grammatical error correction.

Lasertagger proposed by Malmi (Malmi et al., 2019) employs a sequence tagging approach for GEC. It transforms the GEC problem into a text editing task, since the edit distance between an error sentence and its correction is always low. The method mainly combines a BERT encoder with an autoregressive Transformer decoder to predict three main edit operations: keeping a token, deleting a token, and adding a phrase before a token. We use $\mathrm{Lasertagger_{AR}}$ with RoBERTa(Cui et al., 2019) as backbone and set the vocab size to 1000 during the phrase extraction phase, other parameters are set to default.

Conv-Seq2Seq proposed by Gehring(Gehring et al., 2017) is a CNN based sequence to sequence (Seq2Seq) model. It treats the GEC problem as a machine translation task, and translates a noisy, ungrammatical sentence into a clean one. Different from RNN based Seq2Seq models, it is more efficient during the training stage and more sensitive to local n-gram features. Previous work by Ren (Ren et al., 2018) has shown its effectiveness. We use the same setting as Ren during train.

3.4 Spelling Check Models

We also employ spelling check models, SpellGCN and Electra, for better dealing with spelling errors. Note that when applying spell checkers to GEC, we use a post-process strategy to filter false positive samples, including modification of person name, or number or “他他它”, and modification which makes target sentence segment length larger than raw sentence segment length.

SpellGCN(Cheng et al., 2020) divides each similar character set into two similarity graphs, one for similar pronunciation and one

Input: <go> 歌曲使人感到快乐，<eos> Predicted: 歌曲被人很快乐，<eos>
Input: <go> 现在知道你讲中文很风趣了！<eos> Predicted: 今天我知道你的讲中文很风趣！<eos>
Input: <go> 你是哪里人？<eos> Predicted: 你是那里人？<eos>
Input: <go> 手下们都累得气喘吁吁的时候，那位老人的身份被公开了。<eos> Predicted: 下手都累得气喘吁吁的时候，那位老人被身分公开了。<eos>
Input: <go> 我们一起玩了一些时候，然后吃了晚饭。<eos> Predicted: 所以我们一起玩一些时间，然后就吃晚饭。<eos>
Input: <go> 关于中国政府奖学金申请，我有几个问题希望得到您的解答。<eos> Predicted: 有关于中国政府奖学金申请，我有几个问题希望得到您的答应。<eos>
Input: <go> 我害怕如果我们没有进展，那样我会很难过 <eos> Predicted: 我害怕如果我们不可以进展，不过我很难过 <eos>
Input: <go> 考虑到韩国的老年贫困率和老年自杀率位居世界第一这一点，情况就特别严重。<eos> Predicted: 考虑韩国的老年贫困率老年的杀害率位于世界第一这一点，情况特别严重。<eos>
Input: <go> 尽管我有了非常多空闲，但我还没写完我答应要发送给你的故事。<eos> Predicted: 无论我有非常多空，我还没写完我答应要送给你的故事。<eos>

Table 3: Neural network-based corruption results.

for similar shape. Then it takes the graphs as input and generates an embedding for each character after the interaction between similar characters. These embeddings are then constructed into a character classifier for the semantic representation extracted from another backbone module. With the Combination of graph representation and BERT, SpellGCN can leverage the similarity knowledge and generate the right corrections accordingly. We use the default setting as Cheng, and a fine tuned BERT by Xu(Ming, 2020) as the backbone.

Electra(Clark et al., 2020) is a pre-training language model with a new pre-training task and framework, which changed the generative masked language model (MLM) pre-training task into the discriminant replaced token detection (RTD) task to determine whether the current token has been replaced by the language model. Experiments of paper show that the context representation learned by Electra is much better than the context representation learned by Bert and XL-net under the same model size, data, and calculation conditions. We use the Chinese version Electra-base model released by iFLYTEK Joint Laboratory of Harbin Institute of technology for spelling check.

3.5 N-Best Reranker

With the mentioned spelling check models and generative models, we use a recursive method for Chinese GEC as shown in Figure 1. For a given input sentence, it will first go through two spelling check models in a parallel fashion. Then the post spelling check output with the original sentence will go through the generative models, also in parallel. The whole process can loop for K (K=3) rounds to obtain N (N=6) output sentences. A MERT reranker is deployed to rerank N sentences, and we select the top-1 sentence as the correction result. Note that we use a post processing script to transform the result to present the detection and position level result. Several features are introduced during reranking: 1. Normalized 4-gram language model score divided by sentence length, 2. Edit operations including character add/delete/swap count from

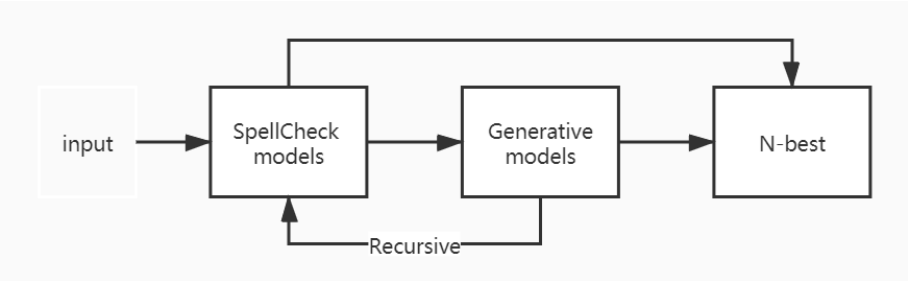


Figure 1: System diagram of our GEC system

source sentence to target sentence, 3. mask-predicted Bert probability score of a target sentence (Chollampatt et al., 2019), 4. Target sentence length penalty. The reranker is based on MERT from Moses (Koehn et al., 2007), with tuning metric of M2 F1-score.

Also, a more simple reranking approach is deployed to choose the best sentence according to the mask-predicted Bert score, which is computed by:

$$\mathrm{S_{BERT}(Y)} = \sum_{\mathrm{i=1}}^{\mathrm{[Y]}} \log \mathrm{P_{BERT}(y_i|Y_{i-masked})}$$

where Y is the target sentence, $Y_{i-msked}$ is the target sentence with i-th word y_i being masked, P_{Bert} is the Bert output probability.

4 Experiments

We first train Lasertagger on HSK+Lang8 dataset to obtain LASER_RAW. Then we add spelling checkers to obtain LASER_RAW+SPELL, which boosts each score by about 0.03. We also train a Lasertagger with augmented data only to obtain LASER_AUG. The promising result shows the effectiveness of our data augmentation methods. After adding spelling checkers, denoted as LASER_AUG+SPELL, we obtain the best correction score, which is 0.1993. Also, we try to pretrain a Lasertagger on augmented data for 10 epoch and fine tune on real data (HSK+Lang8), denoted by LASER_FINE. The model performs better than LASER_RAW and LASER_AUG on detection level and identification level, but worse in correction level. With spelling checkers adding in, we obtain LASER_FINE+SPELL, with a similar boosting effect as previous results. Finally, we add in Conv-Seq2Seq and the two rerankers and denoted as MERT_RERANK and SIMPLE_RERANK. We observe a boosting effect on each level except for a significant downgrade in correction level when using MERT. We submitted the MERT_RERANK as the final result. The testing results are shown in Table 4.

We first train Lasertagger on HSK+Lang8 dataset to obtain LASER_RAW. Then we add spelling checkers to obtain LASER_RAW+SPELL, which boosts each score by about 0.03. We also train a Lasertagger with augmented data only to obtain LASER_AUG. The promising result shows the effectiveness of our data augmentation methods. After adding spelling checkers, denoted as LASER_AUG+SPELL, we obtain the best correction score, which is 0.1993. Also, we try to pretrain a Lasertagger on augmented data for 10 epoch and fine tune on real data (HSK+Lang8), denoted by LASER_FINE. The model performs better than LASER_RAW and LASER_AUG on de-

EXPS	Detection	Identification	Position	Correction Top1
LASER_RAW	0.8258	0.5080	0.2375	0.1332
LASER_RAW+SPELL	0.8517	0.5410	0.2649	0.1668
LASER_AUG	0.8221	0.5319	0.2595	0.1826
LASER_AUG+SPELL	0.8475	0.5672	0.2799	**0.1993**
LASER_FINE	0.8597	0.5610	0.2531	0.1602
LASER_FINE+SPELL	0.8731	0.5837	0.2727	0.1857
MERT_RERANK	**0.8852**	**0.6203**	0.2812	0.1683
SIMPLE_RERANK	0.8846	0.5966	**0.3009**	0.1976

Table 4: Testing results on CGED2020 testing set.

tection level and identification level, but worse in correction level. With spelling checkers adding in, we obtain LASER_FINE+SPELL, with similar boosting effect as previous results. Finally, we add in Conv-Seq2Seq and the two rerankers and denoted as MERT_RERANK and SIMPLE_RERANK. We observe a boosting effect in each level except for a significant down grade in correction level when using MERT. We submitted the MERT_RERANK as final result. Testing results are shown in Table 2.

Since our pipeline does not work well in the FPR track (FPR$\geq$ 0.7068) and performance downgrade is observed, we look into the MERT reranker results. We find that the 4-gram model does not perform well. It tends to give shorter sentences higher scores, and it was trained on news domain data, so domain adaption can be an issue.

5 Conclusion and Future Works

This paper describes our system in the CGED2020 Shared Task Grammatical Error Correction. We explored a scheme by combining data augmentation methods, spelling check methods, and generative grammatical correction methods. We achieved the best recall score and our final result ranked fourth. However, there are still many efforts needed to solve this problem. A lot of improvements can be made to our current model. In the future, we will continue working on this problem. Possible future directions include improving data augmentation methods, finding a better reranking strategy, and finding better measurements for evaluation.

References

Xingyi Cheng, Weidi Xu, Kunlong Chen, Shaohua Jiang, Feng Wang, Taifeng Wang, Wei Chu, and Yuan Qi. 2020. Spellgcn: Incorporating phonological and visual similarities into language models for chinese spelling check.

Shamil Chollampatt, Weiqi Wang, and Hwee Ng. 2019. Cross-sentence grammatical error correction. pages 435–445.

Kevin Clark, Minh Thang Luong, Quoc V Le, and Christopher D Manning. 2020. Electra: Pre-training text encoders as discriminators rather than generators.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for chinese bert.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, W. Che, S. Wang, G. Hu, and T. Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. In *NLP-TEA@ACL*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning.

Philipp Koehn, Hieu Hoang, Alexandra Birch, Chris Callison-Burch, Marcello Federico, Nicola Bertoldi, Brooke Cowan, Wade Shen, Christine Moran, Richard Zens, Chris Dyer,

OndÅ™ej Bojar, Alex Constantin, and Evan Herbst. 2007. Moses: Open source toolkit for statistical machine translation.

Eric Malmi, Sebastian Krause, Sascha Rothe, Daniil Mirylenka, and Aliaksei Severyn. 2019. Encode, tag, realize: High-precision text editing.

Xu Ming. 2020. pycorrector: Text correction tool [software].

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for chinese grammatical error correction. In *CCF International Conference on Natural Language Processing and Chinese Computing*.

Chencheng Wang, Liner Yang, Yun Chen, Yongping Du, and Erhong Yang. 2019. Controllable data synthesis method for grammatical error correction.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Łukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Greg Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation.

Ziang Xie, Guillaume Genthial, Stanley Xie, Andrew Ng, and Dan Jurafsky. 2018. Noising and denoising natural language: Diverse back-translation for grammar correction. pages 619–628.

Yi Yang, Pengjun Xie, J. Tao, Guangwei Xu, L. Li, and S. Luo. 2017. Alibaba at ijcnlp-2017 task 1: Embedding grammatical features into lstms for chinese grammatical error diagnosis task. In *IJCNLP*.

Bo Zheng, W. Che, Jiang Guo, and T. Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. In *NLP-TEA@COLING*.

TMU-NLP System Using BERT-based Pre-trained Model to the NLP-TEA CGED Shared Task 2020

Hongfei Wang and Mamoru Komachi
Tokyo Metropolitan University
wang-hongfei@ed.tmu.ac.jp,komachi@tmu.ac.jp

Abstract

In this paper, we introduce our system for NLPTEA 2020 shared task of Chinese Grammatical Error Diagnosis (CGED). In recent years, pre-trained models have been extensively studied, and several downstream tasks have benefited from their utilization. In this study, we treat the grammar error diagnosis (GED) task as a grammatical error correction (GEC) problem and use a method that incorporates a pre-trained model into an encoder-decoder model to solve this problem.

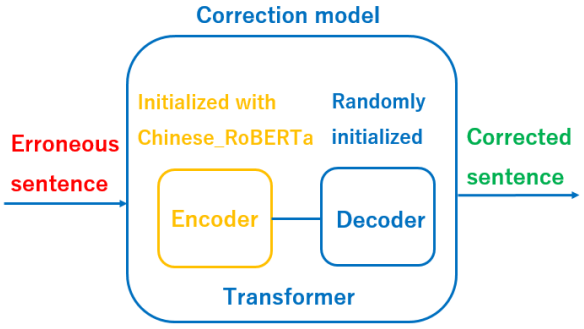


Figure 1: The structure of our system.

1 Introduction

In the CGED task, given a source sentence contains grammatical errors, systems are needed to output four kinds of results: detection level, identification level, position level, and correction level. We treat the CGED task as a GEC problem, so our system will output the corrected sentence directly. Then we use a post-processing method to generate detection, identification, and position level results.

GEC can be regarded as a sequence-to-sequence task. GEC systems receive an erroneous sentence written by a language learner and output the corrected sentence. In previous studies that adopted neural models for Chinese GEC (Ren et al., 2018; Zhou et al., 2018), the performance was improved by initializing the models with a distributed word representation, such as Word2Vec (Mikolov et al., 2013). However, in these methods, only the embedding layer of a pre-trained model was used to initialize the models.

In addition, Chinese GEC remains challenging because Chinese is a complex language. For example, there are more than 10,000 Chinese characters, and the glyphs or pronunciation of several Chinese characters are similar. Therefore, character errors introduced by Chinese learners can be variable; further, for the Chinese GEC tasks, it is necessary to utilize a pre-trained model, which captures additional information about a language. In recent years, pre-trained models based on Bidirectional Encoder Representations from Transformers (BERT) have been studied extensively (Devlin et al., 2019; Liu et al., 2019), and the performance of many downstream Natural Language Processing (NLP) tasks has been dramatically improved by utilizing these pre-trained models. In order to learn existing knowledge of a language, a BERT-based pre-trained model is trained on a large-scale corpus using the encoder of Transformer (Vaswani et al., 2017). Subsequently, for a downstream task, a neural network model is initialized with the weights learned by a pre-trained model that has the same structure and is fine-tuned on training data of the downstream task. The performance is expected to improve by using this two-stage method because downstream tasks are informed by the knowledge learned by the pre-trained model.

In this study, as shown in Figure 1, we develop a Chinese GEC model based on Transformer by initializing the encoder of Transformer with a Chinese BERT-based pre-trained model and then fine-tuning the correction model on Chinese GEC and GED corpus. Our system achieves F_1 scores of 82.00, 52.66, 22.24, 14.17 on detection level, iden-

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 87–90
Suzhou, China, December 4, 2020.

tification level, position level and correction level evaluation.

2 Related Work

In Building Educational Applications (BEA) 2019 (Bryant et al., 2019), several teams attempted to incorporate BERT into their correction models. Kaneko et al. (2019) first fine-tuned BERT on a learner corpus and then incorporated the word probability provided by BERT into re-ranking features. Using BERT for re-ranking features, they obtained an approximately 0.7 point improvement of the $F_{0.5}$ score. Kantor et al. (2019) used BERT to solve the GEC task by iteratively querying BERT as a black box language model. They added a [MASK] token into source sentences and predicted the word represented by the [MASK] token. If the word probability predicted by BERT exceeded the threshold, the word was output as a correction candidate. Using BERT, they obtained a 0.27 point improvement of the $F_{0.5}$ score. These studies show that BERT helps improve the performance of a correction model; however, this improvement was marginal, and they did not explore the use of pre-trained models for weight initialization.

3 Method

We adopt the method from Wang et al. (2020) to construct our correction model. Additional details are introduced in the following sections.

3.1 Chinese Pre-trained Model

We use a BERT-based model as our pre-trained model. BERT is mainly trained with a task called Masked Language Model. In the Masked Language Model task, some tokens in a sentence are replaced with masked tokens ([MASK]), and the model needs to predict the replaced tokens.

In this study, we use the Chinese-RoBERTa-wwm-ext model provided by Cui et al. (2019). The main differences between Chinese-RoBERTa-wwm-ext and original BERT are as follows:

Whole Word Masking (WWM) Devlin et al. (2019) proposed a new masking method called Whole Word Masking (WWM) after proposing their original BERT, which masks entire words instead of subwords. They demonstrated that the original prediction task that only masks subwords is easy and that the performance has been improved by masking entire words. Therefore, Cui et al. (2019) adopted this method to train their Chinese pre-trained models. It should be noted that they used WordPiece (Wu et al., 2016) to preprocess Chinese sentences, and Chinese sentences are segmented into characters (not subwords) by WordPiece. Therefore, in WWM, when a Chinese character is masked, other Chinese characters that belong to the same word should also be masked. Table 1 shows an example of WWM.

[Original Sentence] 然后 **准备** 别 的 **材料** 。
[Original BERT] 然 后 **准 [MASK]** 别 的 **[MASK] 料** 。
[Whole Word Masking] 然 后 **[MASK] [MASK]** 别 的 **[MASK] [MASK]** 。
[English Translation] Then prepare for other materials.

Table 1: Example of the difference between original BERT and Whole Word Masking for Chinese sentences. The original sentence is segmented into words, whereas in original BERT and whole word masking, the sentence is segmented into characters.

Training Strategy Cui et al. (2019) followed the training strategy studied by Liu et al. (2019). Although Cui et al. (2019) referred to the training strategy from Liu et al. (2019), there are still some differences between them (e.g., they did not use dynamic masking).

Training Data In addition to Chinese Wikipedia (0.4B tokens) that was originally used to train BERT, an extended corpus (5.0B tokens), which consists of Baidu Baike (a Chinese encyclopedia) and QA data, was also used. The extended corpus has not been released due to a license issue.

3.2 Grammatical Error Correction Model

In this study, we use Transformer as our correction model. Transformer has shown excellent performance in sequence-to-sequence tasks such as machine translation and has been widely adopted in recent English GEC studies (Kiyono et al., 2019; Junczys-Dowmunt et al., 2018).

However, a BERT-based pre-trained model only uses the encoder of Transformer; therefore, it can not be directly applied to sequence-to-sequence tasks that require both an encoder and a decoder, such as GEC. Hence, we initialize the encoder of Transformer with the parameters learned by Chinese-RoBERTa-wwm-ext, and the decoder is initialized randomly. Finally, we fine-tune this initialized model on Chinese GEC data and use it as our correction model.

Parameters	
Architecture	Encoder (12-layer), Decoder (12-layer)
Learning rate	3×10^{-5}
Batch size	32
Optimizer	Adam
Loss function	cross-entropy
Dropout	0.1

Table 2: Training details for our model.

Evaluation Type	P	R	F_1
Detection	94.04	72.70	82.00
Identification	69.80	42.28	52.66
Position	34.60	16.39	22.24
Correction	22.58	10.32	14.17

Table 3: Experimental results of our model.

4 Experiments

4.1 Experimental Settings

Training Data We use the training data provided by the NLPCC 2018 Grammatical Error Correction shared task. In this task, approximately one million sentences from the language learning website Lang-8[1] are used as training data. We first segment all sentences into characters because the Chinese pre-trained model we used is character-based. In the GEC task, source and target sentences do not tend to change significantly. Considering this, we filter the training data by excluding sentence pairs that meet the following criteria: i) the source sentence is identical to the target sentence; ii) the edit distance between the source sentence and the target sentence is greater than 15; iii) the number of characters of the source sentence or the target sentence exceeds 64. Once the training data are filtered, we obtain 971,318 sentence pairs.

We also use the training data provided by the CGED task this year and in previous years. By dividing the sentence units into separate sentences, we finally obtain 56,000 sentence pairs.

Validation Data We randomly extract 5,000 sentences from the CGED training data as the validation data.

Test Data We use the official test data, which contain 2,456 sentence units. We also divide the sentence units of the test data and obtain 2,745 sentences.

Implementation We implement the Transformer model using fairseq 0.8.0.[2] and load the pre-trained model using pytorch_transformer 2.2.0.[3]

We train our model on the NLPCC training data for 20 epochs and then on CGED training data for another 20 epochs. The first 20 epochs are validated on NLPCC validation data, and the second 20 epochs are validated on CGED validation data.

We select the best model from all 40 epochs according to the loss function (cross-entropy).

We train four models using different random seeds and combine them into a 4-ensemble model.

More details on the training are provided in Table 2.

Evaluation We post-process our system outputs and obtain final outputs.

We first recover the sentence units of test source and system outputs. Then the detection result is decided by whether the output sentence is identical to the source. For identification and position, we use edit-distance to obtain the difference between source and output.

4.2 Evaluation Results

Table 3 summarizes the experimental results of our model. In all 44 submitted system results, our system is 33rd, 33rd, and 31st on detection, identification, and position evaluation. In all 25 results for the correction level, our system is 19th.

4.3 Case Analysis

Table 4 shows the sample outputs.

In the first example, both the gold edit and our system correct the spelling error 果 (fruits) to 课 (class). It appears that our system can accurately correct the error according to context because the word 果 (fruits) rarely comes after the word 汉语 (Chinese).

In the second example, the output of our system is more fluent, although the gold edit does not make any alterations to the source sentence. Our system changes the word 赏玩 into 欣赏, although the two words have a similar meaning: enjoy, a native speaker often uses 欣赏 rather than 赏玩 when the object is 景色 (scenery). It appears that our system can capture the collocation efficiently because the pre-trained model is trained with a large-scale corpus.

In the third example, our system changes the word selection error 殷勤 (attentive) to 勤奋 (working hard), which is irrelated to the context. We

[1] https://lang-8.com/
[2] https://github.com/pytorch/fairseq
[3] https://github.com/huggingface/transformers

src	星期五中午十二点钟我上汉语**果**。	我们可以悠闲地**赏玩**风景。
gold	星期五中午十二点钟我上汉语**课**。	我们可以悠闲地**赏玩**风景。
Our system	星期五中午十二点钟我上汉语**课**。	我们可以悠闲地**欣赏**风景。
Translation	I have a Chinese class at 12 o'clock on Friday.	We can enjoy the scenery leisurely.
src	他知道了我要换房子，因为我的租金太高。所以他请我来跟他一起住以便分担租金。我觉得他很**殷勤**。	
gold	……我觉得他很**善良**。	
Our system	……我觉得他很**勤奋**。	
Translation	He knew that I was going to change house because my rent was too high. So he invited me to live with him in order to share the rent. I think he is very kind.	

Table 4: Source sentence, gold edit, and output of our system.

think that this is because we divide the sentence unit, and so our system can not refer to the context that is necessary to correct this error.

5 Conclusion

In this study, we initialized the encoder of the Transformer with a Chinese pre-trained model and used this system to challenge the CGED task.

References

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 shared task on grammatical error correction. In *BEA@ACL*.

Yiming Cui, Wanxiang Che, Ting Liu, Bing Qin, Ziqing Yang, Shijin Wang, and Guoping Hu. 2019. Pre-training with whole word masking for Chinese BERT. *ArXiv*.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *NAACL-HLT*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *NAACL-HLT*.

Masahiro Kaneko, Kengo Hotate, Satoru Katsumata, and Mamoru Komachi. 2019. TMU transformer system using BERT for re-ranking at BEA 2019 grammatical error correction on restricted track. In *BEA@ACL*.

Yoav Kantor, Yoav Katz, Leshem Choshen, Edo Cohen-Karlik, Naftali Liberman, Assaf Toledo, Amir Menczel, and Noam Slonim. 2019. Learning to combine grammatical error corrections. In *BEA@ACL*.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *EMNLP-IJCNLP*.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke S. Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A robustly optimized BERT pretraining approach. *ArXiv*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *NIPS*.

Hongkai Ren, Liner Yang, and Endong Xun. 2018. A sequence to sequence learning for Chinese grammatical error correction. In *NLPCC*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *NIPS*.

Hongfei Wang, Michiki Kurosawa, Satoru Katsumata, and Mamoru Komachi. 2020. Chinese grammatical correction using BERT-based pre-trained model. In *AACL-IJCNLP*.

Yonghui Wu, Mike Schuster, Zhifeng Chen, Quoc V. Le, Mohammad Norouzi, Wolfgang Macherey, Maxim Krikun, Yuan Cao, Qin Gao, Klaus Macherey, Jeff Klingner, Apurva Shah, Melvin Johnson, Xiaobing Liu, Lukasz Kaiser, Stephan Gouws, Yoshikiyo Kato, Taku Kudo, Hideto Kazawa, Keith Stevens, George Kurian, Nishant Patil, Wei Wang, Cliff Young, Jason Smith, Jason Riesa, Alex Rudnick, Oriol Vinyals, Gregory S. Corrado, Macduff Hughes, and Jeffrey Dean. 2016. Google's neural machine translation system: Bridging the gap between human and machine translation. *ArXiv*.

Junpei Zhou, Chen Li, Hengyou Liu, Zuyi Bao, Guangwei Xu, and Linlin Li. 2018. Chinese grammatical error correction using statistical and neural models. In *NLPCC*.

CYUT Team Chinese Grammatical Error Diagnosis System Report in NLPTEA-2020 CGED Shared Task

Shih-Hung Wu*, Jun-Wei Wang
Chaoyang University of Technology,
Taichung, Taiwan, R.O.C
shwu@cyut.edu.tw, s10827605@gm.cyut.edu.tw
*Contact author

Abstract

This paper reports our Chinese Grammatical Error Diagnosis system in the NLPTEA-2020 CGED shared task. In 2020, we sent two Runs with two approaches. The first one is a combination of conditional random fields (CRF) and a BERT model deep-learning approach. The second one is CRF approach. The official test results shows that our Run1 achieved the highest precision rate 0.9875 with the lowest false positive rate 0.0163 on detection, while Run2 gives a more balanced performance.

1 Introduction

Learning Chinese is very popular for foreigners, but it is difficult for them to write correct sentence. Grammatical error detection is a big challenge for the Chinese learners as a second language. Learning Chinese sentences will rely too much on the teacher to correct the wrong sentences. It is not easy for learners to get timely feedback. Therefore, how to use existing technology to detect and correct the grammatical errors that learners make has become a hot topic.

Since 2014 (Yu et al.,2014) (Lee et al. 2015) (Lee et al. 2016) (Rao et al., 2017) (Rao et al., 2018), the NLP-TEA workshop provides a series Chinese Grammar Error Detection (CGED) shared tasks to promote the research on grammar error diagnosis. The organizers ask professional teachers to label the errors in learners' sentences. There are four types of label in the sentences: Redundant (R), Selection (S), Disorder (W), and Missing (M). The goal of the task is to build a system that can predict whether a sentence is wrong and correct it. In previous years, we participated in the NLPTEA CGED (Wu et al., 2018) and shows that such a system can be precision oriented or recall oriented for different users.

Since the emerging of deep learning, we find that sequence-to-sequence models have good effect on grammar correction, and the BERT model (Devlin et al., 2019; Xu et al., 2019) is the best sequence-to-sequence pre-training language model using a large number of data sets. The pre-trained model is trained with mask language model (MLM) to enhance the strength of the model.

In Run1 of 2020, we use BERT as the first level of our identification. We fine-tuning the BERT model with the Lang-8[1] corpus and all the data from NLPTEA since 2016 to 2020, so that the model can be used to predict correct and incorrect sentences, and reproduce the wrong sentences. The error types are determined by CRF. In Run2 is not used to determine the wrong and correct sentences. In the following sections, we will introduce related work and our approaches, then discuss the formal test results, and give conclusion and future works.

2 Related Work

Grammar error detection and correction is now a popular research topic in natural language processing (Li et al., 2018; Fu et al., 2018). Previous works show that CRF model can be used to integrate various features to build a good system. Better results can be achieved by using the pre-collected collocation word database.

Recently, researchers use deep learning models to solve this issue. The most common models are sequence-to-sequence (Ge et al., 2018) and convolutional neural network (Li et al., 2019) models. The idea of sequence-to-sequence is to translate the wrong into correct sentences just like translation between two languages. A corrected

[1] https://lang-8.com/

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 91–96
Suzhou, China, December 4, 2020.

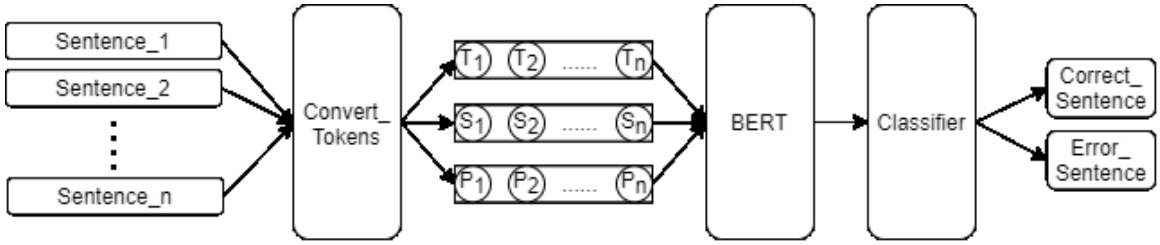


Figure 1 Fine-tune system architecture

sentence is generated from the wrong sentence, and it is believed that multiple revisions will give better results. The convolutional neural network originally is used to process images, now shifted to process text. With the two-dimension processing power, it is easier for the model to read the context of the text.

Since 2019, the BERT gives many state-of-the-art results on several NLP applications. This shows the great influence of BERT on natural language processing. Spelling check is a similar task to the grammar correction (Cheng et al., 2020; Zhang et al., 2020). The authors use the BERT internal model to find typos. Although its effect is not the best, it achieves the purpose the author wants.

3 Method

This year, we mainly focused on minimizing the false alarm on error detection. Since the system is to help foreign learners, we hope that less errors judged by the model will not cause learners to feel frustration.

BERT is a pre-trained language model. Since the original pre-training model was not trained for Chinese grammar correction, we have to train it with our corpus. For different task, better results can be achieved by fine-tuning the pre-trained language models with additional training corpus. Moreover, BERT has achieved excellent results in various projects, such as single classification tasks, sentence-labelling tasks, and question answering tasks. In addition to the BERT model, we also use conditional random fields (CRF) to double check tine wrong sentences detected by BERT, and select the type and location of the errors.

3.1 Fine-tune Language model

We use the BERT pre-training language model provided by huggingface[2]. The pre-train model is "bert-base-chinese". The fine-tuning training data set is the Lang-8 data set provided by NLPCC and all the training data and test data from 2016 to 2020 provided by NLPTEA excepting 2020 test data. Figure 1 shows the BERT fine-tune system architecture. The data set {Sentence _1, Sentence _2, …, Sentence _n} has been preprocessed. Our system compares the original sentence and the modified sentence from the data set. If the sentence is wrong, mark it as "Error_ Sentence ", otherwise mark it as "Correct_ Sentence ". Given a source token = $\{T_1, T_2, ..., T_n\}$ with its segment = $\{S_1, S_2, ..., S_n\}$ and position = $\{P_1, P_2, ..., P_n\}$, we can fine-tune the BERT and obtain the classify results. After classifying the correct and error sentences, the next step the error sentences need input the Conditional Random Fields (CRF). Table 1 shows the number of wrong and correct sentences and their average length in the fine-tuning data set.

	Quantity	Average length
Correct sentence	1,241,126	21
Error sentence	1,117,577	20

Table 1: Fine-tuning data set statistics

3.2 Conditional Random Fields

We use CRF model in both two Runs. Run1 uses a pre-trained language model + CRF and Run2 uses only CRF. We want to see what changes will happen if the pre-trained language model is added. CRF is used to mark the error type and location.

CRF is regarded as a sequence label model. As show in Figure 2, the model will be trained according to the sequence label S we provide, and

[2] https://huggingface.co/transformers/model_doc/bart.html

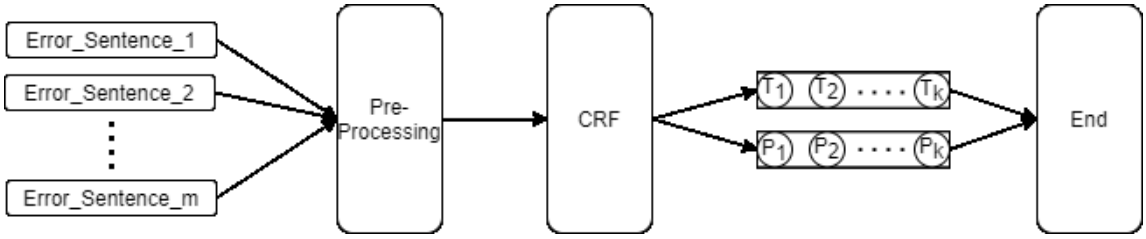


Figure 3 CRF system architecture

the trained model is used to predict the corresponding sequence label Y. The sequence tags we provide to the model contain the words and parts of speech that have been hyphenated by Jieba. The part of speech (POS) can have a very good effect during training. In the first column of the sequence, we place the hyphenated words and the second column. The part of speech of the word, and the label of the wrong type { T_1, T_2, …., T_n} and the position of the word {P_1, P_2, …. , P_n}. Finally, the error type and location are transformed into the format specified by the seminar

3.3 Pre-processing

Figure 3 shows the pre-processing flowchart. The Lang-8 and NLPTEA data are used for fine-tuning the pre-train language model. The sentence before correction must be regarded as an error and the sentence after correction is correct. When preparing the dataset for CRF, our system compares the Lang-8 sentences before and after correction using Jieba segmentation and edit distance. The differences between the two sentences will then be used to determine the three different error types and positions within the edit distance. With the help of Jieba, our system can extract the words in the original sentence and obtain the part-of-speech (POS) tag. The error types include redundant words (R), word selection errors (S), and missing words (M). Next we use the three methods in edit distance. In these methods, insert means missing words, delete means redundant words, and replace means word selection errors. Calculate the position of the wrong word through three ways of editing distance.

We bypass the word ordering errors (W) here because it is very difficult and the training data is too little. The different training materials of NLPTEA and Lang-8 have been marked with error types and positions.

During CRF training, because using too much training data will lead to poor training results, only 57,386 error sentences are used during training.

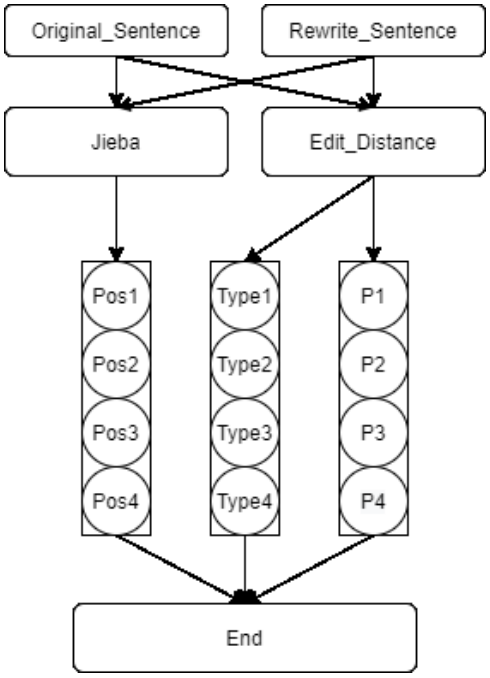


Figure 2 Pre-processing system architecture

Submission	False Positive Rate (the lower the better)
Run1	**0.0163**
Run2	0.5472
Average of 43 runs	0.3920

Table 2: False positive rate in CGED 2020

	Detection Level		
	precision	**Recall**	**F1**
Run1	**0.9875**	0.3443	0.5106
Run2	0.8117	0.6296	0.7091
Average of all 43 runs	0.8922	0.7828	0.8234

Table 3: Detection level in CGED 2020

Each sentence will be processed through Jieba[3] for word segmentation, and finally the corresponding type will be placed in the corresponding position.

4. Experiment Results

4.1 Official test result

Table 2 and Table 3 shows the official results of our system in CGED 2020 shared task. The result shows that our Run1 achieved the highest precision rate 0.9875 with the best false positive rate 0.0163 on detection. In Run2 we improved the recall greatly from 0.3443 to 0.6296 with a drop at the precision rate from 0.9875 to 0.8117. The trade-off of precision and recall is still obvious.

4.2 Error analysis

When encountering sentences that are too long. Our model cannot predict the correct result very well. Here we think that in the fine-tuned training set, it can be seen that the average length is only 20-21. However, as shown in Table 5, the average length of sentences judged by BERT are all above 38 and only a few are below 38. So in the future, we will try to use GPT2 or GPT3 to detect errors in long sentences. Table 4 shows some examples that includes errors but our BERT system fails to detect. As shown in Table 6, we can see the number of errors for the three types of errors. The most numerous are all dependent on one word. Error types R and S almost have similar errors including "的", "是" and "了" and so on. The error type M is mostly punctuation. Because most people usually filter out punctuation because of the convenience of training. Punctuation can make a bad article easier to read. In the future work, we will modify the model towards the above problems.

5. Discussion and Conclusion

In 2020 NLP-TEA CGED shared task, we submitted two Runs, the result shows that our Run1 achieved the highest precision rate (0.9875) with the lowest false positive rate (0.0163) on error detection. The result shows that our system can point out errors with very a high confidence. With very low false alarm, the system can help learners to notice that they really make a grammar error. However, the recall rate of our system is only not very high in Run1. In Run2 we improved the recall greatly from 0.3443 to 0.6296 with a drop at the precision rate from 0.9875 to 0.8117. The trade-off of precision and recall needs more attention.

In the future, we will combine the methods of BERT and GPT2 to improve sentences that our current system cannot detect effectively. About the correction level, we also hope to filter out the best alternative words through GPT2's sentence rewriting method.

# or Sentence	Average length
460	38

Table 5: The sentence statistics of test set

Example	Sentences	Length
1	一个月干下来，大山看上去都没有什么变化。愚公不理会嘲笑，带着全家，继续搬。	37
2	一个兵人暗想：“我要做这个东西了，不然我要被征罚了。”他不暗想：“我不要这样行动因为是不过的”。	48
3	旅行营会把所有的景点、交通和住宿都安排好了。所以自己不能决定哪里去。而且参加旅行营会跟很多不认识的人一起旅行。碰到讨厌的人就没办法，一定要跟他们在一起。所以我喜欢自己一个人旅行。这一样就自由多了。	98
4	星期一上午在大学上课。星期二下牛跟同学一起打排球。	25
5	我有三个哥哥。我最小年轻了。我爸爸妈妈住在法国巴黎。我爸爸是日内瓦大学的老师，所以他很忙。但是，他只教每周只教三天。我妈妈是研究员。她每天上班，所以更忙。	77

Table 4: Examples of long sentences with grammar errors in CGED 2020

[3] https://github.com/fxsjy/jieba

R	R_num	S	S_num	M	M_num
的	433	的	265	。	362
了	308	而	77	，	224
是	198	个	68	不	144
在	98	有	56	的	133
有	82	做	55	一	113
上	63	在	52	我	96
也	51	得	48	是	75
我	43	是	44	有	74
对	42	對	38	很	71
而	42	也	37	人	54
要	39	对	37	这	46
会	37	了	36	在	45

Table 6. NlPTEA CGED 2016 – 2020 Testset, the most frequent errors in each error type

Acknowledgments

This study was supported by the Ministry of Science and Technology under the grant number MOST 109-2221-E-324-024.

References

Xingyi Cheng; Weidi Xu; Kunlong Chen; Shaohua Jiang; Feng Wang; Taifeng Wang; Wei Chu; Yuan Qi, SpellGCN: Incorporating Phonological and Visual Similarities into Language Models for Chinese Spelling Check, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, p.p 871–881, July 5 - 10, 2020.

Jacob Devlin; Ming-Wei Chang; Kenton Lee; Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019.

Ruiji Fu; Zhengqi Pei; Jiefu Gong; Wei Song; Dechuan Teng; Wanxiang Che; Shijin Wang; Guoping Hu; Ting Liu, Chinese Grammatical Error Diagnosis using Statistical and Prior Knowledge driven Features with Probabilistic Ensemble Enhancement, in Proceedings of The 5th Workshop on Natural Language Processing Techniques for Educational Applications, Melbourne, Australia, July 19, 2018.

Tao Ge; Furu Wei; Ming Zhou, Fluency Boost Learning and Inference for Neural Grammatical Error Correction, in Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, p.p 1-11, Melbourne, Australia, July 15 – 20, 2018.

Lung-Hao Lee, Gaoqi Rao,Liang-Chih Yu, Li-Ping Chang, Xun Endong, Baolin Zhang, Li-Ping Chang. 2016. Overview of the NLP-TEA 2016 Shared Task for Chinese Grammatical Error Diagnosis. 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA'16), Osaka, Japan.

Lung-Hao Lee, Liang-Chih Yu, and Li-Ping Chang. 2015. Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis. In Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications (NLP-TEA'15), pages 1-6, Beijing, China.

Chen Li; Junpei Zhou; Zuyi Bao; Hengyou Liu; Guangwei Xu; Linlin Li, A Hybrid System for Chinese Grammatical Error Diagnosis and Correction, in Proceedings of The 5th Workshop on Natural Language Processing Techniques for Educational Applications, Melbourne, Australia, July 19, 2018.

Si Li; Jianbo Zhao; Guirong Shi; Yuanguap Tan; Huifang Xu; Guang Chen; Haibo Lan; Zhiqing Lin, Chinese Grammatical Error Correction Based on Convolutional Sequence to Sequence Model, IEEE Access., vol. 7, pp. 72905 - 72913, May 17, 2019. doi: 10.1109/ACCESS.2019.2917631

Gaoqi Rao, Baolin Zhang, Endong Xun. 2017. IJCNLP-2017 Task1: Chinese Grammatical Error Diagnosis. 8th International Joint Conference of Nature Language Processing (IJCNLP2017), Taipei, Taiwan.

Gaoqi Rao, Qi Gong, Baolin Zhang, Endong Xun, 2018. Overview of NLPTEA-2018 Share Task Chinese Grammatical Error Diagnosis, in Proceedings of The 5th Workshop on Natural Language Processing Techniques for Educational Applications, Melbourne, Australia, July 19, 2018.

Shih-Hung Wu; Jun-Wei Wang; Liang-Pu Chen; Ping-Che Yang, CYUT-III Team Chinese Grammatical Error Diagnosis System Report in NLPTEA-2018 CGED Shared Task, in Proceedings of The 5th Workshop on Natural Language Processing Techniques for Educational Applications, Melbourne, Australia, July 19, 2018.

Hu Xu; Bing Liu; Lei Shu; Philip S. Yu, BERT Post-Training for Review Reading Comprehension and Aspect-based Sentiment Analysis, arXiv:1904.02232v2,May 4, 2019.

Liang-Chih Yu, Lung-Hao Lee, and Li-Ping Chang (2014). Overview of Grammatical Error Diagnosis for Learning Chinese as a Foreign Language. Proceedings of the 1st Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA'14), Nara, Japan, 30 November, 2014, pp. 42-47.

Shaohua Zhang; Haoran Huang; Jicong Liu; Hang Li, Spelling Error Correction with Soft-Masked BERT, in Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, pp. 882–890, July 5 - 10, 2020.

Chinese Grammatical Error Diagnosis Based on RoBERTa-BiLSTM-CRF Model

Yingjie Han [1,2], Yingjie Yan[1,2], Yangchao Han[1], Rui Chao[1], Hongying Zan [1,2]

[1]School of Information Engineering, Zhengzhou University, Zhengzhou Henan, China
[2]Zhengzhou Zoneyet Technology Co., Ltd., Zhengzhou Henan, China
ieyjhan@zzu.edu.cn, yjyan@gs.zzu.edu.cn, hanyangchao@foxmail.com
zzuruichao@163.com, iehyzan@zzu.edu.cn

Abstract

Chinese Grammatical Error Diagnosis (CGED) is a natural language processing task for the NLPTEA6 workshop. The goal of this task is to automatically diagnose grammatical errors in Chinese sentences written by L2 learners. This paper proposes a RoBERTa-BiLSTM-CRF model to detect grammatical errors in sentences. Firstly, RoBERTa model is used to obtain word vectors. Secondly, word vectors are input into BiLSTM layer to learn context features. Last, CRF layer without hand-craft features work for processing the output by BiLSTM. The optimal global sequences are obtained according to state transition matrix of CRF and adjacent labels of training data. In experiments, the result of RoBERTa-CRF model and ERNIE-BiLSTM-CRF model are compared, and the impacts of parameters of the models and the testing datasets are analyzed. In terms of evaluation results, our recall score of RoBERTa-BiLSTM-CRF ranks fourth at the detection level.

1 Introduction

The number of foreigners learning Chinese is constantly increasing. Some foreign countries even regard Chinese as their second language. Learners of Chinese as a foreign language (CFL) may make grammatical errors in writing Chinese. And the goal of Chinese grammatical error diagnosis (CGED) shared task is to develop NLP techniques to automatically diagnose grammatical errors in Chinese sentences written by L2 learners. Such errors fall into four categories: redundant words (denoted as a capital "R"), missing words ("M"), word selection errors ("S"), and word ordering errors ("W").

The criteria for judging correctness are determined at three levels as follows. (1) Detection-level: to distinguish whether a sentence contains grammatical errors; (2) Identification-level: to identify the types of those errors type; (3) Position-level: to detect positions where errors occur. The quality of diagnosis is measured by FPR (False Positive Rate), Pre (Precision), Rec (Recall), and F1.

CGED shared task has been held since 2014 (YUa et al.,2014). In CGED of NLP-TEA 2018 (Rao et al.,2018), deep learning models are widely used, LSTM-CRF has been a standard implementation (Fu et al.,2018; Zhou et al., 2018). While, in recent years, pre-training models, such as BERT, XLNET, ERNIE(Sun et al.,2019) and RoBERTa (Liu et al., 2019) achieve good performance in various NLP tasks (Qiu et al.,2020) because of their fast convergence speed and less cost.

This paper proposes a RoBERTa-BiLSTM-CRF model to detect grammatical errors. The model is described as follows:

(1) The RoBERTa model contains general domain data features and fine-tunes the CGED training data to obtain the corresponding word vectors.
(2) The BiLSTM layer captures sentence-level features based on the powerful long-term memory ability, and CRF works for adjusting labels. The CRF layer only learns from word information without any handcraft features.
(3) In this CGED shared task, our model is only used to detect grammatical errors but not correct them.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 97–101
Suzhou, China, December 4, 2020.

2 Models

We regard the CGED task as a sequence labeling task. The illustrative graph of RoBERTa-BiLSTM-CRF is shown in Figure 1. Chinese characters are input into RoBERTa，and RoBERTa converts each character into a one-dimensional vector. Vector T_1, T_2, …T_n fused with semantic features are output.

The BiLSTM layer makes full use of the context information of the input sequence in the sequence labeling task so that it can predict label more accurately.

The CRF layer fully considers the context correlation when predicting the label. More importantly, the Viterbi algorithm of CRF uses the dynamic programming method to find the path with the highest probability. Therefore, it fits better with the task of CGED and avoids illegal sequences, such as 'B-R' tag followed by 'I-R' tag.

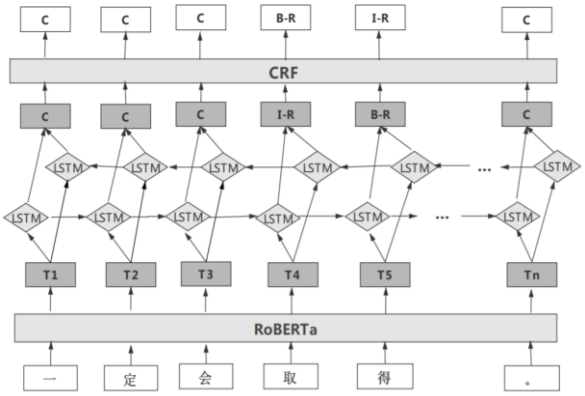


Figure 1: A RoBERTa-BiLSTM-CRF model

2.1 RoBERTa model

RoBERTa model can represent relationships between various words and extract important features in the text. The transformer structure of RoBERTa can get vector representations of sentences from inputting tokens. The RoBERTa model uses a dynamic mask strategy, the model will gradually adapt to different mask strategies processing continuous input data. Compared with training ERNIE model，training RoBERTa model needs larger data sizes and batches. Besides, RoBERTa-large has more network layers and a more complex structure.

2.2 BiLSTM layer

BiLSTM (Bidirectional Long-Short Term Memory) model is composed of a forward LSTM (Long-Short Term Memory) model and a backward LSTM model (Hochreiter et al.,1997). Each word contains information from forward and backward at any time. LSTM model remembers or forgets previous information through the internal gate structure: forgetting gate, memory cell, input gate, and output gate. Figure 2 shows a basic unit of LSTM.

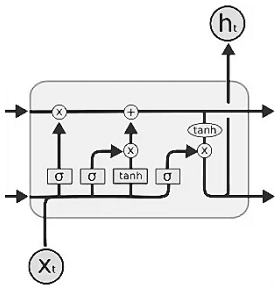


Figure 2: Basic unit in LSTM, it contains forgetting gate, memory cell, input gate and output gate.

1) Forgetting gate as shown in formula (1) selects information to forget, in which h_{t-1} indicates the previous moment; X_t indicates input words, and f_t indicates the output of forgetting data.

$$f_t = (W_t \cdot [h_{t-1}, X_t] + b_f) \quad (1)$$

2) Memory gate selects information to remember, as shown in formula (2), in which i_t indicates the output of the memory gate, and C_t indicates the temporary cell's state shown in formula (3).

$$i_t = (W_i \cdot [h_{t-1}, X_t] + b_i) \quad (2)$$

$$C_t = tanh(W_c \cdot [h_{t-1}, X_t] + b_c) \quad (3)$$

3) Memory cell records cell state C_t in the current moment, as shown in formula (4). The last cell state is C_{t-1}.

$$C_t = f_t \cdot C_{t-1} + i_t \cdot C_t \quad (4)$$

4) Formula (5) and (6) show output gate result O_t and state of this moment h_t.

$$O_t = W_o \cdot [h_{t-1}, X_t] + b_o \quad (5)$$

$$h_t = O_t \cdot \tanh(C_t) \quad (6)$$

2.3 CRF layer

The last layer CRF (conditional random field) (Lafferty et al., 2001) are used to learn an optimal path (Liu et al., 2018). The output dimension of the Bi-LSTM layer is tag size, and the score of input

sequence X corresponds to the output tag sequence y is defined as formula (7).

$$s(X, y) = \sum_{i=0}^{n} A_{y_i, y_{i+1}} + \sum_{i=1}^{n} P_{i, y_i} \quad (7)$$

P represents an output matrix of Bi-LSTM, where P_{ij} represents the non-normalized probability of word w_i mapped to tag_j, and A_{ij} represents the transition probability of tag_i to tag_j.

Softmax function work for defining a probability value $P(y|X)$ as shown in formular (8) for each correct tag sequence y.

$$P(y|X) = \frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}} \quad (8)$$

In training, maximizing the likelihood probability $P(y|x)$. Therefore, we define the loss function as $-log(P(y|X))$, and then use the gradient descent method to learn the network. It is shown in formula (9).

$$\log\big(p(y|X)\big) = \log\left(\frac{e^{s(X,y)}}{\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}}\right) = \mathrm{S(X,y)} - \log\big(\sum_{\tilde{y} \in Y_X} e^{s(X,\tilde{y})}\big) \quad (9)$$

3 Dataset

We collect training datasets of CGED2016 (HSK) (Lee et.al, 2016), CGED2017 (Rao et.al, 2017), CGED2018, and CGED2020 as training dataset and validation dataset, with a total of 21938 data units. The ratio of training dataset size to validation dataset size is about 8:2. We adopt the CGED2018 testing dataset as our experimental testing dataset, with a total of 3549 data units. CGED2020 testing dataset has a total of 1457 data units. Table1 shows the number of data units, the number of errors grouped by error types in the training dataset, validation dataset, test2018, and test2020.

	Training dataset	**Validation dataset**	**Test 2018**	**Test 2020**
Units	17461	4476	3541	1457
Errors	42335	10583	5040	3595
R	9507	2377	1119	768
M	10963	2741	1381	816
S	19072	4768	2167	1688
W	3157	789	373	323

Table 1: The number of data units, number of errors and distributions of error types in training dataset, validation dataset, test2018, and test2020.

We segment sentences into separate characters, and tag label for every character. Label 'C' indicates correct character; 'B-X' indicates the beginning position for an error of type 'X' and 'I-X' shows the middle or ending position for an error of type 'X'. Eight kinds of labels in our data: 'B-R', 'I-R', 'B-M', 'B-S', 'I-S', 'B-W', 'I-W', and 'C'. The sample of processed data is shown in Table 2.

Original data format:
<DOC>
<TEXT id="200505109525100098_2_9x1">
即使父母好好指导孩子，如果父母每天玩的话，对孩子的效果也没有。
</TEXT>
<CORRECTION>
即使父母好好指导孩子，如果父母每天玩的话，对孩子的教育效果也没有。
</CORRECTION>
<ERROR start_off="26" end_off="26" type="M"></ERROR>
<ERROR start_off="25" end_off="25" type="R"></ERROR>
<ERROR start_off="26" end_off="30" type="W"></ERROR>
</DOC>

Processed data format:
即 C\n 使 C\n 父 C\n 母 C\n 好 C\n 好 C\n 指 C\n 导 C\n 孩 C\n 子 C\n， C\n 如 C\n 果 C\n 父 C\n 母 C\n 每 C\n 天 C\n 玩 C\n 的 C\n 话 C\n，C\n 对 C\n 孩 C\n 子 C\n 的 B-R\n 效 I-W\n 果 I-W\n 也 I-W\n 没 I-W\n 有 I-W \n。C\n

Table 2: A data unit sample of original data and processed data, every character has a label.

4 Experiments

4.1 Experimental results and discussions

In the shared task, RoBERTa-BiLSTM-CRF model (Model1) and RoBERTa-CRF model (Model2) are used. Different epochs are set on Model1 and the general parameters of two models are shown below:

- Learning rate 1e-5
- Batch size 16
- Embedding size 1024
- Hidden size 128
- Max length 100

Methods		Model1(epoch=50)	Model2(epoch=50)	Model1(epoch=60)
False Positive Rate		**0.5265**	0.722	0.6933
Detection-level	**Pre.**	0.6817	0.6247	0.6355
	Rec.	0.8896	0.9481	0.9536
	F1	**0.7719**	0.7532	0.7627
Identification-level	**Pre.**	0.5553	0.5274	0.5564
	Rec.	0.5802	0.6412	0.6513
	F1	0.5675	0.5689	**0.6001**
Position-level	**Pre.**	0.3108	0.3078	0.4389
	Rec.	0.2946	0.3129	0.4287
	F1	0.3025	0.3103	**0.4337**

Table 3：Results of three experiments (two models) at three levels on test2018. Model1 represents for RoBERTa-BiLSTM-CRF model, and Model2 for RoBERTa-CRF model

Methods		Run1	Run2	Run3
False Positive Rate		0.8708	0.7557	**0.6938**
Detection-level	**Pre.**	0.8118	0.8182	0.8254
	Rec.	0.9304	0.9078	0.8757
	F1	**0.8671**	0.8607	0.8498
Identification-level	**Pre.**	0.5899	0.6150	0.64
	Rec.	0.5126	0.5076	0.5214
	F1	0.5485	0.5562	**0.5746**
Position-level	**Pre.**	0.29	0.2874	0.2783
	Rec.	0.1941	0.1892	0.2042
	F1	0.2326	0.2282	**0.2356**

Table 4： Results of three runs submitted in shared CGED task. Model2(epoch=50), Model1 and Model2(epoch=60) on test2020.

The following metrics at detection-level, identification-level, and position-level are Pre, Rec, F1, besides an integrated FPR. The results of Model1 (epoch=50; epoch=60) and Model2 on test2018 are shown in Table 3.

F1 scores of Model1 are higher than Model2 at detection-level but lower than Model2 at identification-level and position-level. Since the BiLSTM model learns the dependency relationship between sentences, Model1 may capture error information accurately from the global sequences.

F1 scores of models with larger epoch at identification-level and position-level are higher. This is because larger epoch may lead to overfitting of Model1 at detection-level but not at identification- level and position-level.

Table 4 shows the three runs submitted to the CGED2020 shared task. Run1 is based on the Model1 with 50 epochs; Run2 is based on Model2 with 50 epochs, and Run3 is based on Model1 with 60 epochs.

In this shared task, we get a good recall score of Model1 at the detection-level with bad FPR score. The reason may be as follows. The training corpus of the pre-training model, which comes from news, community discussions, and encyclopedias, is different from the training dataset of CGED, and may easily recognize correct sentences as sentences with grammatical errors.

The performances of three runs on test2020 are consistent with that on test2018 in sum. But F1 scores of three runs on test2020 at detection-level are all higher than that of test2018. According to statistics of errors in Table 1, a data unit contains an average of 1.4233 errors on test2018, while a data unit contains an average of 2.467 errors on test2020. This may lead to diagnosis models more easily to predict whether a sentence contains grammatical errors or not.

4.2 Follow-up experiments and discussions

After the CGED2020-TEA, we use ERNIE-BiLSTM-CRF model (Model3) to do this task. F1 score of Model1 and Model3 on test2018 and test2020 can be seen in Table 5 and Table 6. Model1 gets a worse performance than Model3 at three levels on test2018 but better performance on test2020. The reason is that RoBERTa includes 24 transformers, 16 attention head, and 1024 hidden layer units, which make the generalization ability of RoBERTa-BiLSTM-CRF strong.

	Model1	**Model3**
Detection-level	0.7719	**0.7755**
Identification-level	0.5675	**0.6138**
Position-level	0.3025	**0.4451**

Table 5： F1 scores of Model1 and Model3 on test2018

	Model1	**Model3**
Detection-level	**0.8671**	0.8311
Identification-level	**0.5485**	0.527
Position-level	**0.2326**	0.2153

Table 6： F1 scores of Model1 and Model3 on test2020

5 Conclusion and Future work

This paper proposes a RoBERTa-BiLSTM-CRF model to detect grammatical errors in the CGED shared task. The results of experiments show RoBERTa-BiLSTM-CRF is a good model for detecting grammatical errors in general since RoBERTa model obtains word vector according to data feature, and BiLSTM-CRF captures sentence-level features to predict and adjust labels. In the three runs submitted, our recall ranks fourth at detection- level in the CGED shared task.

In addition, we find that the performance of ERNIE-BiLSTM-CRF is unreasonable on test2020 in our experiments, we will try to pursue reasons from model structure and characters of datasets in the future work.

Acknowledgments

Special thanks to the organizers of CGED 2020 for their great job. We also thank the anonymous reviewers for insightful comments and suggestions.

References

Fu, Ruiji, et al. Chinese grammatical error diagnosis using statistical and prior knowledge driven features with probabilistic ensemble enhancement. *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018.

Hochreiter, Sepp, and Jürgen Schmidhuber. Long short-term memory. *Neural computation* 9.8 (1997): 1735-1780.

Lafferty, John, Andrew McCallum, and Fernando CN Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. (2001).

Lee, Lung-Hao, Liang-Chih Yu, and Li-Ping Chang. Overview of the NLP-TEA 2015 shared task for Chinese grammatical error diagnosis. *Proceedings of the 2nd Workshop on Natural Language Processing Techniques for Educational Applications*. 2015.

Liu, Yinhan, et al. Roberta: A robustly optimized bert pretraining approach. *arXiv preprint arXiv:1907.11692* (2019).

Liu, Yajun, et al. Detecting simultaneously Chinese grammar errors based on a BiLSTM-CRF model. *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018.

Qiu, Xipeng, et al. Pre-trained models for natural language processing: A survey. *arXiv preprint arXiv:2003.08271* (2020).

Rao, Gaoqi, et al. IJCNLP-2017 task 1: Chinese grammatical error diagnosis. *Proceedings of the IJCNLP 2017, Shared Tasks*. 2017.

Rao, Gaoqi, et al. Overview of NLPTEA-2018 share task Chinese grammatical error diagnosis. *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018.

Sun, Yu, et al. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

YUa, Liang-Chih, Lung-Hao LEE, and Li-Ping CHANG. Overview of Grammatical Error Diagnosis for Learning Chinese as a Foreign Language.

Zhou, Yujie, Yinan Shao, and Yong Zhou. Chinese Grammatical Error Diagnosis Based on CRF and LSTM-CRF model. *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*. 2018.

Chinese Grammatical Errors Diagnosis System Based on BERT at NLPTEA-2020 CGED Shared Task

Hongying Zan[1,2], **Yangchao Han**[1], **Haotian Huang**[1], **Yingjie Yan**[1,2], **Yuke Wang**[1], **Yingjie Han**[1,2]

[1]School of Information Engineering, Zhengzhou University, Zhengzhou Henan, China
[2]Zhengzhou Zoneyet Technology Co., Ltd.
{iehyzan,ieyjhan}@zzu.edu.cn, hanyangchao@foxmail.com
grenouillehuang@gmail.com, yjyan@gs.zzu.edu.cn, ykwangyoko@163.com

Abstract

In the process of learning Chinese, second language learners may have various grammatical errors due to the negative transfer of native language. This paper describes our submission to the NLPTEA 2020 shared task on CGED. We present a hybrid system that utilizes both detection and correction stages. The detection stage is a sequential labelling model based on BiLSTM-CRF and BERT contextual word representation. The correction stage is a hybrid model based on the n-gram and Seq2Seq. Without adding additional features and external data, the BERT contextual word representation can effectively improve the performance metrics of Chinese grammatical error detection and correction.

1 Introduction

With the improvement of China's international status, more and more foreigners begin to learn Chinese. Unlike English, Chinese grammar lacks morphology and singular and plural changes, and its sentence patterns are flexible and changeable. In learning Chinese, foreigners are prone to introduce grammatical errors due to the complexity of Chinese itself, the negative transfer of mother tongue and target language, and the cultural differences of different countries.

In order to promote the development of automatic detection of syntactic errors in Chinese writing, the Natural Language Processing Techniques for Educational Applications(NLP-TEA) have taken CGED as one of the shared tasks since 2014. Thanks to the CGED task, some research achievements have been made in Chinese grammatical error detection. Based on those previous research results, this paper puts forward a new thinking direction for the CGED task. Some typical examples are shown in Table 1:

TEXT:他们知不道吸烟对未成年年会造成的各种害处。
GED:<3,4,W>,<12,12,S>,<22,23,S>
GEC :他们不知道吸烟对未成年人会造成的各种伤害。

Table1: Typical error example of CGED dataset

CGED has four subtasks:

(1) Detection-level: Binary classification of a given sentence, that is, correct or incorrect, should be completely identical with the gold standard. All error types will be regarded as incorrect.

(2) Identification-level: This level could be considered as a multi-class categorization problem. All error types should be clearly identified. A correct case should be completely identical with the gold standard of the given error type.

(3) Position-level: In addition to identifying the error types, this level also judges the occurrence range of the grammatical error. That is to say, the system results should be perfectly identical with the quadruples of the gold standard.

(4) Correction-level: For the error types of Selection and Missing, recommended corrections are required. At most 3 recommended corrections are allowed for each S and M type error. In this level, the amount of the corrections recommended would need influent the precision and F1 in this level. The trust of the recommendation would be tested.

This paper is organized as follows: Section 2 describes some related works in English and Chinese grammar error diagnosis. Section 3 introduces the hybrid system that we proposed.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 102–107
Suzhou, China, December 4, 2020.

Section 4 shows the evaluation and discussion of our system. Section 5 concludes the paper and discusses future work.

2 Related Work

The automatic diagnosis of grammatical errors is a topic of natural language processing. More research on the task of automatic grammatical error recognition focuses on English. In the 1960s, the study of automatic proofreading of English texts was carried out abroad. The HOO (Helping Our Own) (2011) task related to grammatical errors in the task are all about English, which attracts many English grammatical errors researchers. Researchers have proposed a variety of technologies suitable for automatic detection and correction of English grammatical errors, such as rule-based methods (Foster et al., 2004), phrase-based statistical methods (Gamon., 2010), machine learning-based methods (Rei et al., 2016).

However, there are few studies on grammatical errors in modern Chinese. Starting in 2014, the Natural Language Processing Techniques for Educational Applications (NLPTEA) has added modern Chinese grammatical error recognition tasks. These evaluations The task provides a good platform for researchers to showcase their work, and it also speeds up the progress of modern Chinese grammatical errors in automatic recognition methods. At different stages of the development of science and technology, the research methods of modern Chinese grammatical error recognition are different, from rule-based to statistics-based, and then to deep learning-based methods. Zheng (2016) proposed a model based on stacked LSTM and CRF in 2016, which improved the accuracy and recall rate of automatic grammatical error recognition. In the 2017 IJCNLP-2017 CGED evaluation, Yang (2017) proposed a sequence labelling model based on BiLSTM−CRF, which combines the establishment of parts of speech, n-gram grammar, and dependency features, and uses multiple model results to merge and delete After the last 20% of the results are merged, and the results are voted three different integration mechanisms, the effect of automatic grammatical error recognition has been dramatically improved in the F1 value of the three levels, Fu(2018) in the 2018 NLPTEA-2018 CGED evaluation task. Based on the BiLSTM-CRF model, it combines new features such as Gaussian point-by-point mutual information, and adopts multiple model results for probabilistic integration and mixed multiple results ranking. Two different integration mechanisms are introduced in the post-processing.

GEC is typically formulated as a sentence correction task. A GEC system takes a potentially erroneous sentence as input and is expected to transform it into its corrected version. The CoNLL-2014 shared task test set is the most widely used dataset to benchmark GEC systems. The test set contains 1,312 English sentences with error annotations by two expert annotators. Models are evaluated with the MaxMatch scorer, which computes a span-based Fβ-score.

In the NLPCC2018-task2-CGEC (Zhao et al., 2018), the You Dao team (Fu et al., 2018) regards the error correction task as a translation task. Errors are divided into surface errors and grammatical errors. The similar phonetic table and 5-gram language model are used to solve low-level errors, and the Transformer model based on character granularity and word granularity are used to solve high-level errors. Combine the low-level model and the high-level model and finally use the 5-gram language model to analyze the corrected sentence's perplexity and select the sentence with the lowest perplexity. The Ali team(Zhou et al., 2018) adopts a multi-model parallel structure, using three types of models: rule-based, statistics-based, and neural network. First, the low-level combination, which includes one rule based model, two SMT based models, and four NMT based models, obtains the category candidates, and then the high-level combination merges the candidates generated by the low-level combination.

3 System Description

The system proposed in this paper contains two parts: the error detection stage and the error correction stage. The hybrid model presented in this paper is shown in Figure 1.

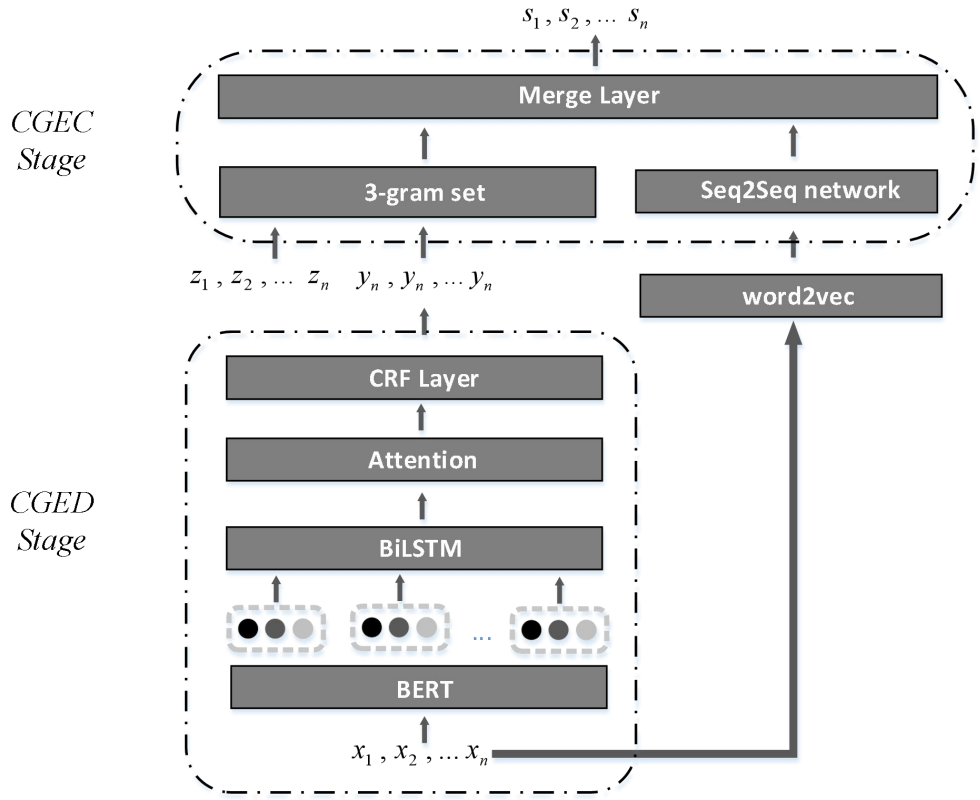


Figure 1: The structure of our system.

First, in the error detection stage, we integrate the BERT method and attention mechanism on the traditional BiLSTM-CRF model. The input is a sequence of characters $\{x_1,x_2,...x_n\}$. The output is the dynamic word vector sequence $\{w_1,w_2,...w_n\}$, after encoding layer and decoding layer, we can get the label sequence $\{y_1,y_2,...y_n\}$ .Then, in the error correction stage, we perform 3-gram extraction based on the corrected sentence sequence $\{z_1,z_2,...z_n\}$ of CGED2016-2018, and construct a quadruple with frequency information. According to the results obtained by the detection stage, we will extract the label sequences containing M or S, merge the error-checking results with the rewrite results of seq2seq, and obtain the final result information $\{s_1,s_2,...s_n\}$.

3.1 Detection Stage

The experimental training data set in this article is a CGED training set that integrates 2016-2018, and the test set is the CGED 2018 test set. First, we need to preprocess the data set. Set the label set to {C, R, M, S, W} to indicate no grammatical error, R type error, M type error, S type error, W type error. According to the grammatical error information marked in the data set, each word is marked with the corresponding label. The processed form is: char, word / POS / dependency / label. The processed data is input into the model for experimentation. After deleting 114 units without control, 21937 units are left for training. Our training set statistics are shown in Table 2.

Training Set	units	errors
CGED2016	10071	24797
CGED2017	10449	26448
CGED2018	402	1067
CGED 2020	1129	2909
Sum	22051	55221
Invalid Data	114	203
Using Data	21937	55018

Table 2: Training set statistics

Example of sentences before processing is shown as follows:

<DOC>
<TEXT id="20030710952310053 8_2_4x1">
农作物也是不例外。
</TEXT>
<CORRECTION>
农作物也不例外。
</CORRECTION>
<ERROR start_off="5" end_off="5" type="R">
</ERROR>

The example of preprocessed data is shown in Table 3.

Char	Word	POS	DEP	Label
农	农作物	B-n	B-SBV	C
作	农作物	I-n	I-SBV	C
物	农作物	I-n	I-SBV	C
也	也	B-d	B-ADV	C
是	是	B-v	B-HED	B-R
不	不	B-d	B-ADV	C
例	例外	B-v	B-VOB	C
外	例外	I-v	I-VOB	C
。	。	B-wp	B-WP	C

Table 3: The example of preprocessed data

BERT embedding layer: The semantic information between sentence sequences in the traditional model is extracted by BiLSTM. The vectors of words in the embedding layer are the same in different semantic environments. This may confuse the semantic information of the sentence. BERT uses a two-way Transformer structure. Transformer uses a multi-head attention mechanism, each layer has the same structure but different weights, each layer focuses on different features, and the overall feature is obtained. It can learn the contextual relationship between texts by paying attention to important information between sequences. Since the model does not pay attention to the sequence order, the position is introduced Information features to strengthen the extraction of location information, making it a deeper understanding of the context. The input is a sequence of characters $\{x_1,x_2,...x_n\}$,through the BERT neural network, the beginning of each sentence is marked by [CLS], and the mark [SEP] is added to the end of each sentence, which means that a sentence embedding is added to each In terms of characters, token embedding, sentence embedding, and transformer position embedding respectively represent character vectors, sentence vectors, and position vectors. In Chinese grammatical error recognition. In the model, the model input is a single sentence. Add a position embedding to each character to indicate its position in the sequence. The output is the dynamic word vector sequence $\{w_1,w_2,...w_n\}$ after BRRT encoding, which is input into the encoding module as a word vector feature.

Encoding layer: The encoding module uses BiLSTM. The input of this module is a sequence of dynamic word vectors encoded by BERT, which can be expressed as $\{w_1,w_2,...w_n\}$. BiLSTM generates the hidden state sequence corresponding to each character by encoding the character vector. The bidirectional LSTM obtains the forward and backward hidden states by reading the sequence from left to right and reading the sequence information from right to left, respectively. The layer output is the splicing of the front and backs hidden states, and the output hidden layer sequence is $\{h_1,h_2,...h_n\}$.

Decoding layer: The decoding module uses BiLSTM-CRF, and at the same time adds an attention mechanism. Although BiLSTM extracts contextual information, there is no correlation between the output sequences. It only predicts the optimal at each moment. In order to capture useful information for the error recognition task, an attention mechanism is added to give different information obtained by decoding. Attention weight. CRF uses transition features to constrain the output sequence and output the final predicted label sequence, where, represents the set of all predicted labels. The output of the CRF layer is the final predicted label, the label set is {C, R, M, S, W}, and each word in the input sequence is labeled with a corresponding label.

3.2 Correction Stage

The experimental data set of the error correction part uses the data set of NLPCC2018-TASK2, which has a total of 717241 sentence pairs. After deleting 123501 data sets without grammatical error and 513 data sets without control, 593227 data sets are left. This paper divides the data set and conducts experiments according to the ratio 10000:10000:573227 of test set, validation set, and training set.

Seq2seq model: This article uses the encoding method for each Chinese character, that is, the sentence is converted into a sequence of Chinese characters, and the Chinese characters are encoded by character vectors, and the word2vec character vector is adopted, and the character vector dimension is 200 dimensions. Input the word vector into seq2seq for training. The optimizer uses rmsprop, and the loss function uses cross entropy. After 40 rounds of training, the model can output the corrected sentence well. The output of the model is shown in Table 4:

Input sentence: 请 把 我 修 改 一 下 ！
Decoded sentence: 请 帮 我 修 改 一 下 ！
Target sentence: 请 帮 我 修 改 一 下

Table 4: The example of seq2seq model's prediction

n-gram model: We extracted 400,490 3-gram combinations from 20,000 correct sentences through the NLTK tool. If the previous word and the next word in the error position are the same as the beginning and end of the triple, the middle word will be the recommended word. Then the model will use the frequency of 3-gram appearance as the answer score, sort according to the score and get the best answer.

4 Experiment Results

CGED evaluation indicators include false positive rate, accuracy, precision, recall rate, F1 value, in order to evaluate the performance of the system at the four levels of grammatical errors.

$$False\ Positive\ Rate = FP/(FP + TN) \quad (1)$$

$$Accuracy = TP + TN/(TP + FP + FN + TN) \quad (2)$$

$$Precision = TP/(TP + FP) \quad (3)$$

$$Recall = TP/(TP + FN) \quad (4)$$

$$F1 = 2 * P * R/(P + R) \quad (5)$$

Table 5 shows the experiments results that the system BERT-BiLSTM-CRF+Correction model performs best among many models. This is because BERT encodes sentences, effectively extracts the dynamic word vector features of sentences, and adds an attention mechanism to the decoding. It further extracts meaningful information from the decoded tags and improves the correction effect.

We also find out that our result didn't perform well in FPR. Because the CGED task belongs to the cost unequal experiment, we should try to increase the cost of marking the sentences with non-error type as error in the experiment instead of treating the cost as the same.

Methods		**BERT-BiLSTM-Attention-CRF+Correction (epoch=100)**	**Char/Word/POS/DEP+BiLSTM-CRF+Correction (epoch=100)**	**Char/Word+BiLSTM-CRF (epoch=100)**
False Positive Rate		**0.6645**	0.6775	0.7394
Detection-level	**Pre.**	**0.8262**	0.8145	0.8136
	Rec.	0.8435	0.7939	**0.8617**
	F1	0.8348	0.8041	**0.8370**
Identification-level	**Pre.**	**0.5856**	0.5053	0.5018
	Rec.	0.4416	0.4127	**0.5060**
	F1	0.5035	0.4543	**0.5039**
Position-level	**Pre.**	**0.2502**	0.0996	0.067
	Rec.	**0.1472**	0.0665	0.0613
	F1	**0.1854**	0.0798	0.0640
Correction-level	**Pre.**	**0.0027**	0.0009	
	Rec.	**0.0012**	0.0004	
	F1	**0.0017**	0.0006	

Table 5: Results on the test data

	Detection Level		
	Precision	**Recall**	**F1**
Run1	0. 8262	**0. 8435**	**0. 8348**
Average of 43runs	0. 89	0. 78	0. 82

Table 6: Performance evaluation in Detection Level

The performance of our hybrid system is shown in the following tables comparing to the average of all 43 formal runs in 2020. Table 6 shows our metrics on detection level. As we expected, BERT-BiLSTM-CRF+Correction model gives the perform well in both recall and F1.

5 Conclusion

Aiming at the problems of the traditional models for automatic recognition of grammatical errors in Chinese, such as the complex features and the large number of model integrations that are difficult to train, this paper proposes a BERT-BiLSTM-Attention-CRF+Correction model that combines the BERT word vector and attention mechanism. Compare it with the multi-feature BiLSTM-CRF and CRF models. The experimental results on the 2020 NLPTEA evaluation data set show that the BERT-BiLSTM-Attention-CRF model performs better than other models we submitted, proving the superiority of BERT word vectors in feature representation.

On the basis of the model proposed in this article, comparing the effects of embedding different pre-trained word vectors on the recognition effect, and how to add a large amount of external knowledge to the recognition model to improve performance are issues worth exploring in our future work.

References

Bo Zheng, Wanxiang Che, Jiang Guo, and Ting Liu. 2016. Chinese grammatical error diagnosis with long short-term memory networks. *In Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications (NLPTEA2016)*, pages 49–56.

Fu, K.,Huang, J.,& Duan, Y. 2018. Youdao's winning solution to the nlpcc-2018 task 2 challenge: a neural machine translation approach to Chinese grammatical error correction. *Lecture Notes in Computer Science*, *vol 11108*. Springer, Cham. Pages 341-350. https://doi.org/10.1007/978-3-319-99495-6_29

Jennifer Foster and Carl Vogel. 2004. Parsing ill-formed text using an error grammar. *Artificial Intelligence Review*, 21(3-4):269–291.

Marek Rei and Helen Yannakoudakis. 2016. *Compositional sequence labelling models for error detection in learner writing. arXiv preprint* arXiv:1607.06153.

Michael Gamon. 2010. Using mostly native data to correct errors in learners′ writing: a meta-classifier approach. *In Human Language Technologies: The 2010 Annual Conference of the North American Chapter of the Association for Computational Linguistic*s, pages 163 − 171. Association for Computational Linguistics.

Dale, Robert, and Adam Kilgarriff. 2011, September. elping our own: The HOO 2011 pilot shared task. *In Proceedings of the 13th European Workshop on Natural Language Generation*. pp242-249.

Ruiji Fu, Zhengqi Pei, Jiefu Gong, Wei Song, Dechuan Teng, Wanxiang Che, Shijin Wang, Guoping Hu,and Ting Liu. 2018. Chinese grammatical error diagnosis using statistical and prior knowledge riven features with probabilistic ensemble enhancement. *In Proceedings of the 5th Workshop on atural Language Processing Techniques for Educational Applications*, pages 52–59.

Yi Yang, Pengjun Xie, Jun Tao, Guangwei Xu, Linlin Li, and Luo Si. 2017. Alibaba at ijcnlp-2017 task 1: Embedding grammatical features into lstms for Chinese grammatical error diagnosis task. *In Proceedings of the IJCNLP 2017, Shared Tasks*, pages 41–46.

Zhao Y, Jiang N, Sun W, & Wan X. 2018. Overview of the NLPCC 2018 Shared Task: Grammatical Error Correction: 7th CCF International Conference, NLPCC 2018, Proceedings, Part II. *Natural Language Processing and Chinese Computing. Springer*,pages439-445. https://doi.org/10.1007/978-3-319-99501-4_41

Zhou J., Li C., Liu H., Bao Z., Xu G., Li L. 2018.Chinese Grammatical Error Correction Using Statistical and Neural Models. In: Zhang M., Ng V., Zhao D., Li S., Zan H. *Natural Language Processing and Chinese Computing*. NLPCC 2018. Lecture Notes in Computer Science, vol 11109. Springer, Cham. https://doi.org/10.1007/978-3-319-99501-4_10

Chinese Grammatical Error Detection Based on BERT Model

Yong Cheng Mofan Duan

Faculty of Arts. , Ludong University.

Abstract

Automatic grammatical error correction is of great value in assisting second language writing. In 2020, the shared task for Chinese grammatical error diagnosis(CGED) was held in NLP-TEA. As the LDU team, we participated the competition and submitted the final results. Our work mainly focused on grammatical error detection, that is, to judge whether a sentence contains grammatical errors. We used the BERT pre-trained model for binary classification, and we achieve 0.0391 in FPR track, ranking the second in all teams. In error detection track, the accuracy, recall and F-1 of our submitted result are 0.9851, 0.7496 and 0.8514 respectively.

1 Introduction

With the development on economy and international influence of China,, more and more foreigners begin to learn Chinese. However, Chinese language is one of the most complex and difficult languages in the world, which is hard to master for foreigners. As a result, foreigners may produce a lot of grammatical errors in their written compositions which involve various error types such as Word Redundant Error, Word Missing Error, Word Selection Error, and Word Disorder Error (Gaoqi, 2018). So It has become an important task and challenge for TCFL teachers to help foreign students to detect, understand and correct these grammatical errors.

In recent years, natural language processing technology has been developing rapidly, and the latest deep learning technology has promoted the overall development of artificial intelligence greatly（LeCun, 2015; Schmidhuber, 2015）. In this background, computer-aided education has received more and more attention in NLP area, and one of the representative work is automatic chinese grammatical error diagnosis. In this task, computers are trained to detect and correct the grammatical errors in the compositions written by foreigners. And this work can provide a pretty assist on teaching second language writing.

In 2020, the evaluation on Chinese Grammatical Error Diagnosis (CGED) is held as a shared task in NLPTEA workshop. CGED evaluation has been held for six times, which has greatly promoted the development of related technologies. We participated the CGED 2020 as team LDU. We submitted three runs and achieved the second place in the FPR track. In this paper we will introduce our work in detail.

2 Related Work

Early research on grammar error correction mainly focused on English. And the rule-based approach was popular in early research **(**Michael, 2008; Gabor, 2013**)**, For example, the grammar checker in Mircrosoft Word is a broad-coverage rule-based proofreading system. However, this system was designed for native speakers, and cannot detect errors in texts written by second language learners（Claudia, 2014）. With the development of machine learning technology, the data-driven based approach has become the gold standard method in the field of grammatical error correction（William, 1992; Na, 2006; Kevin, 1994）. Since 2011, four shared tasks are held to evaluate grammar error correction technology, that is the 2011 HOO shared task, the 2012 HOO shared task, the 2013 CONLL shared task, and the 2014 shared task **(**Robert, 2011&2012; Hwee, 2013&2014**),** These shared tasks has greatly promoted the

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 108–113
Suzhou, China, December 4, 2020.

development of English grammar error correction approaches. And these approches mainly focus on particular errors such as Article errors, prepositional errors and so on. In the recent 2019 Bea shared task (Christopher, 2019), the focus of the research has moved to the correction of the whole sentence. Meanwhile, the Transformer-based machine translation neural network model has been widely adoptesd in error correction task, which greatly improve the state of the art（Yo, 2019).

In contrast, the research on Chinese grammatical error correction started relatively late. One major driving force in this area is the Shared Task of Chinese Grammatical Error Diagnosis (CGED) organized by Beijing Language and Culture University. It has been held for six times from 2014 to 2020 (Liang, 2014; Lung, 2015-2016; Gaoqi, 2017-2018). The goal of the CGED is to identify the types and positions of grammatical errors in sentences and correct them. In previous studies, the mainstream method is to treat the error correction task as a sequence annotation task, and LSTM, CRF and other models to used to diagnose the error (Chen, 2018). In addition, NLPCC has also held an shared task in 2018 **(**Yuanyuan, 2018**)**, which forcus on the whole sentence correction task, and the sequence to sequence neural network model has been widely used in the participating teams (Kai 2018.).

3 Our Approach

3.1 Task Description

The goal of CGED shared task is to develop NLP techniques to automatically diagnose grammatical errors in Chinese sentences written by Chinese as Foreign Language (CFL) learners. Four error types are defined as redundant words (denoted as a capital “R”), missing words (“M”), word selection errors (“S”), and word ordering errors (“W”). one or more such errors may occur in the input sentences. The developed system should indicate whether the sentence contain any errors, If the inputs contain no grammatical errors, the system should return ”correct”. Otherwhise, the error types and the position in sentence should be returned, and the system can provide their word for the missing word in error “M” and the wrong word in error “S”, Figure1 show the input and system output.

In the evaluation stage, five tracks were set up, including false positive, error detection, error type, error location, S & M error correction. The evaluation results and final ranking were given

INPUT: 即使父母好好指导孩子，如果父母每天玩的话，对孩子的效果也没有。
OUTPUT:
Error1: 26, 26 Missing(M)
Error2: 25, 25 Redundant (R)
Error3: 26, 30 Disorder (W)

INPUT: 现在必须得考虑怎样对待这种社会问题了。
OUTPUT: Correct

Figure 1. eamples of system input and output.

respectively according the tracks. Our team LDU focuses on the first two tracks, false positives and error detection. Which is to detect whether a sentence contains grammatical errors. We regard the error detection task as basic problem of grammatical error correction, so it is worthy of in-depth study. In this next, we will introduce our datasets and models.

3.2 Datasets

In our work, we use three different datasets: HSK dynamic composition corpus (HSK Dataset), language8 corpus (Lang8 Dataset), and primary and secondary school error corpus (School Dataset). We will introduce these three types of corpora respectively.

(1) HSK Dataset

"HSK dynamic composition corpus" (Endong, 2018) is a corpus collected from the HSK advanced writing test for foreigners whose mother tongue is not Chinese. This corpus was collected by Beijing Language and Culture University which contain the composition and corresponding answers of some foreign candidates from 1992 to 2005. The total number of the composition is 11569, with a total of 4.24 million words. The corpus was manually annotated with the error types and corrections from different text levels such as word level, sentence level, and paragraph level. Most of the training sets used by CGED are from the HSK Dataset.

(2) Lang8 Dataset

Lang8 Dataset **(**Yuanyuan, 2018**)** is collected from http://lang-8.com/, a language-learning website where native speakers freely choose learners’ essays to correct. This dataSet are used in the Grammatical Error Correction shared task in NLPCC2018. They collect a large-scale Chinese Mandarin learners’ corpus by exploring “language exchange” social networking services (SNS).

There are about 68,500 Chinese Mandarin learners on this SNS website. By collecting their essays written in Chinese and the revised version by Chinese natives, we set up an initial corpus of 1,108,907 sentences from 135,754 essays.

(3) School Dataset

School dataset is collected from the homework books, composition books, weekly notebooks, diaries, examination papers and so on of the students in primary and secondary schools by Ludong University. The collected data was then processed and annotated manually according to the error types. The total number of records of the corpus is 100631, and the total number of words is more than 3 million and it's all from native Chinese speaker. Dataset not open source for now since it's may not completed.

3.3 Our Model

In this paper, we regard the grammatical error detection task as a binary classification problem to judge whether a sentence contains grammatical errors by training a classification model. On data preprocessing, we transform the dataset D into the form of pairs< T, S>, where S denotes the sentece, and T denote the tags which contains "correct" and "wrong". And In training stage, we adapt the BERT pre-trained model and fine tuning method to train the classification model.

The Bert model (Devlin, 2018) was proposed by Google in 2018. Its full name is Bidirectional Encoder Representation from Transformers. This neural network contains three layers: embedding layer, transformer layer and prediction layer, as in Fig 2. In embedding layer, the input consists of three parts: token embedding, segment embedding and position embedding. The function of transformer layer is to encode the input information. Different from convolutional neural network and recurrent neural network, the main characteristic of the transformer architecture is the use of self attention mechanism to mine and search the hidden relations within text sequences, which can effectively improve the performance of various sequence tasks. Finally, in the prediction stage, the model contains two subtasks, one is to predict the relationship between sentences and the other is to predict the cover words. Specifically, the Bert model extracts two sentences A and B from the dataset, in which the probability of sentence B being the next sentence of sentence A is 50%, and 15% of the words in the two sentences are randomly masked. Then the information is input into the embedding layer and the transformer layer for encoding. Finally, the output of the transformer layer is used to predict the hidden words in sentences A and B, and the probability that sentence B is the next sentence of sentence A.

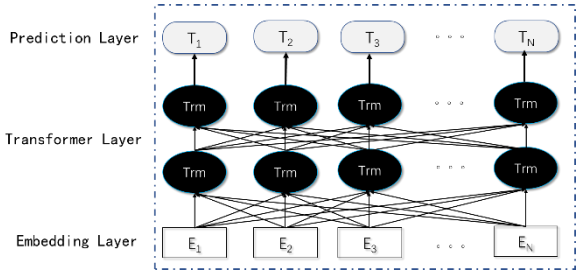


Figure2. Architecture of BERT model

In this paper, we adopt three different pre-training models based on BERT, which named Bert-base (Devlin, 2018), Roberta（Yinhan, 2019) and Roberta wwm（Yiming, 2019). And we train the classification model by fine tuning on the grammatical error corpus. On this basis, the label category of the sentence is predicted. The overall structure of our work is shown in the Figure 3.

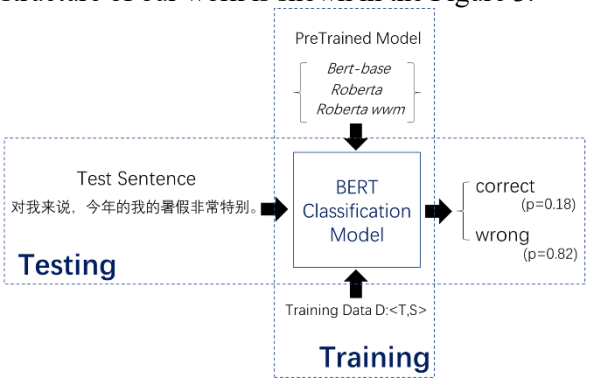


Figure 3. Structure of our work

4 Experiment

4.1 Experiment Setting

As for data, we use four different datasets. The first dataset is provided by CGED committee, in which all the data from 2014 to 2018 are used as training data, and the training set and testing set from 2020 are used as the validation set and the test set in our experiment. The other three datasets are HSK dataset (excluding the duplicate items with CGED dataset), school dataset and lang8 data set. The scale of these four datasets are shown in the Table 1.

Dataset	CGED dataset	HSK dataset	School dataset	Lang8 dataset
Number (sens)	91,967	202,671	25,951	1,786,290

Table 1. Scale of four datasets

Our work focuses on the task of grammar error detection, and we use False Positive Rate (FPR)、Precision(P)、Recall(R) and F-Measure(F-1) as our evaluating indicator.

4.2 Experiment Result

(1) Single Model Comparison

In this paper, we tried three kinds of pre-trained Bert models, named Bert_base, Roberta and Roberta_wwm. in addition, we compare these models with several baseline models based on the TF-IDF features, we use the following classification models: Gauss naive Bayes (gnb), random forest (rf). The results are shown in Table 2.

Result on Validation Set		FPR	P	R	F-1
Base Line	gnb	0.53	0.519	0.571	0.544
	rf	0.568	0.526	0.632	0.574
BERT	BERT-Base	0.134	0.833	0.667	0.741
	RoBERTa	0.131	0.837	0.675	0.747
	RoBERTa -wwm	0.136	0.835	0.688	0.754
Result on Test Set		FPR	P	R	F-1
Base Line	gnb	0.536	0.506	0.550	0.527
	rf	0.604	0.507	0.621	0.558
BERT	BERT-Base	0.052	0.98	0.685	0.807
	RoBERTa	0.065	0.975	0.684	0.804
	RoBERTa -wwm	0.062	0.977	0.691	0.810

Table 2. Experiment result on different models

It can be seen that compared with baseline model, the model based on Bert has a significant improvement in both the validation set and the test set. When comparing different pre-trained models, we can see that the performance of Roberta_wwm is higher than the other two. The FPR and F-1 on verification set are 0.136 and 0.754, and 0.062 and 0.810 on test set. In general, the performance on the test set is better than that on the verification set. It's shown in the first submitted run of LDU.

(2) Experiment on Sample Proportion

In this part, we adjust the proportion of "corret" samples and "wrong" samples in the training set, based on which we can see the influence of the proportion of positive and negative samples on the results. The number of positive and negative samples of the CGED dataset is shown in the table 3. It can be seen that the proportion on training set and validation set is close to 1:1, while the proportion on test set is close to 4:1.

	Training Set	Valid Set	Test Set
Wrong number	44536	1129	1150
Correct number	43716	1129	307

Table 3. Experiment result on sample proportion

On this basis, we adjust the proportion of "correct" and "wrong" samples in the training set by reduce the proportion of correct samples and wrong samples by 10%, 30%, 50%, 70%, 90% respectively, noted as c-10%, c-30%, c-50%, c-70%, c-90%, w-10%, w-30%, w-50%, w-70%, w-90%. On this basis, we use the best Roberta wwm model to carry out experiments on the verification set and test set, the result is shown in Figure 4.

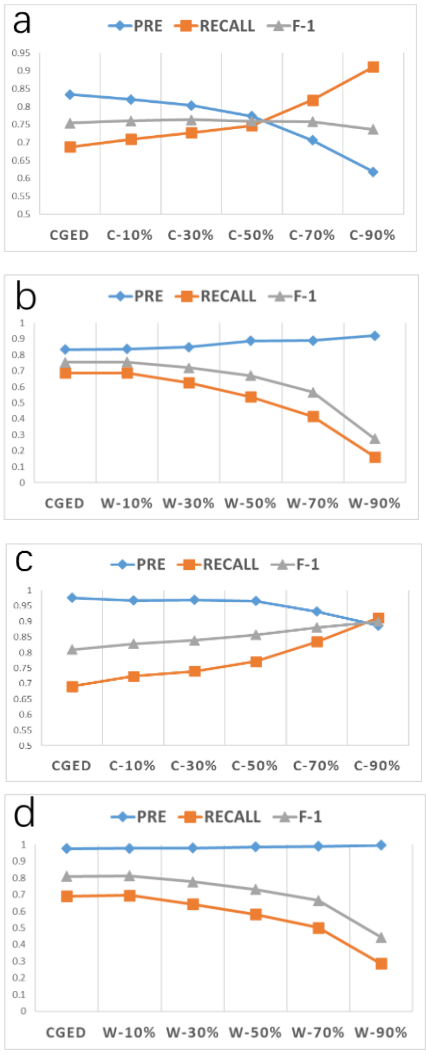


Figure 4. Result on different proportion.(a) Correct samples reduction on validation set. (b) Wrong samples reduction on validation set. (c) Correct samples

reduction on test set. (d) Wrong samples reduction on test set.

From the above results, we can see that when the proportion of "wrong" samples is reduced in both validation set and test set, the precision increases slightly, while the recall decreases rapidly, so the overall F-1 also decreases. However, when the proportion of "correct" samples is reduced, the results are different between validation set and test set. In validation set, The precision decrease rapidly, while the recall increase rapidly, so the overall F-1 value is relatively stable. While in test set, it is obvious that the precision decreases slightly, but both the recall and F-1 value increase rapidly. When the proportion of "correct" lables are reduced to 90%, the F-1 value reaches about 0.9, which is close to the highest level in all the participating teams. For the performance change, We think it is related to the high proportion of false tags in the test set. Therefore, when the proportion of false tags in the training set increases, the overall performance on the test set will increase. It can be seen that the proportion of "correct" and "wrong" samples in the training set has a great impact on the grmmar error detection performance. It's shown in the second submitted run of LDU.

(3) Experiment on Data Augmentation

In this part, we try the data augmentation experiment, three other dataset(hsk, school, lang8) in table are added to the original CGED training set, namely CGED+hsk, CGED+school, CGED+lang8, and we conducted experiments on the verification set and test set respectively, the result is shown in Table 4.

	Valid Set			Test set		
	P	R	F-1	P	R	F-1
CGED	0.835	0.688	0.754	0.977	0.691	0.810
CGED +hsk	0.942	0.702	0.804	0.993	0.624	0.766
CGED +school	0.761	0.791	0.776	0.953	0.815	0.879
CGED +lang8	0.871	0.268	0.410	0.977	0.262	0.413

Table 4. Experiment result on data augmentation

It can be seen that when hsk data is added, the performance on the verification set increases greatly, while the performance on the test set decreases. We believe that the difference between the verification set and the test set is mainly caused by the proportion difference of "correct" and "wrong" samples. When the school data is added, the performance increase slightly both on on the verification set and test set. We think that the CGED test set may contain some homologous errors that primary and secondary school students may also make, so increasing the school data will improve the overall performance. When the lang8 data is added, the performance on the verification set and the test set decrease rapidly. We think lang8 dataset itself has a large number of noise, which is quite different from cged data, so the performance decreases. It's shown in the third submitted run of LDU.

5 Conclusion

This is a technical report of our LDU team participating in CGED2020 shared task. We mainly participated in the task of grammar error detection. We regard the task as a binary classification task, and use different Bert pre-trained models and datasets in our experiments. The experimental results show that the BERT model can greatly improve the accuracy of grammar error detection. And we conclude that the homogeneity and the smaples proportion distribution in training set and test set have a great impact on the final performance.

Acknowledgments

This paper is supported by the research project of Chinese dictionary research center of the state language and writing Commission:"A Study on the Construction of a Hierarchical Retrieval System for Multi-source Sample Sentences for Lexicography Media", whose project number is CSZX-YB-202004.

References

Gaoqi R, Qi G, Baolin Z and Endong X. 2018. Overview of NLPTEA-2018 Share Task Chinese Grammatical Error Diagnosis. Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications, pages 42–51.

LeCun Y, Bengio Y, Hinton G. 2015. Deep Learning. Nature 521(7553):436-44.

Schmidhuber J. Stockmeyer. 2015. Deep learning in neural networks: An overview. Neural Networks, 6: 85-117.

Michael G, Jianfeng G, Chris B, et al. 2008. Using contextual speller techniques and language modeling for ESL error correction. In Proceedings of the International Joint Conference on Natural Language Processing (IJCNLP), pages 449–456.

Gabor B, Veronika V, Sina Z, et al. 2013. LFG-based features for noun number and article grammatical errors. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, pages 62–67.

Claudia L, Martin C, Michael G and Joel. 2014. T. Automated Grammatical Error Detection for Language Learners, Second Edition. Published by Morgan & Claypool.

William G, Ken C and David Y. 1992. Work on statistical methods for word sense disambiguation. In Proceedings of the AAAI Fall Symposium on Probabilistic Approaches to Natural Language, pages 54–60.

Na R, Martin C and Claudia L. 2006. Detecting errors in English article usage by non-native speakers. Natural Language Engineering, 12(2):115–129.

Kevin K and Ishwar C. 1994. Automated postediting of documents. In Proceedings of the Twelfth National Conference on Artificial Intelligence (AAAI), pages 779–784.

Robert D and Adam K. 2011. Helping Our Own: Text massaging for computational linguistics as a new shared task. in Sixth International Natural Language Generation Conference.

Robert D, llya A and George N. 2012. HOO 2012: A report on the preposition and determiner error correction shared task. In Proceedings of the Seventh Workshop on Building Educational Applications Using NLP, pages 54–62.

Hwee T, Siew M, Yuanbin W, et al. 2013. The CoNLL2013 Shared Task on Grammatical Error Correction. In Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task, pages 1–12.

Hwee T, Siew M, Ted B, et al. 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.

Christopher B, Mariano F, et al. 2019. The BEA-2019 Shared Task on Grammatical Error Correction. Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 52–75.

Yo J, Jiyeon H, Kyubyong P and Yeoil Yoon. 2019. A Neural Grammatical Error Correction System Built On Better Pre-training and Sequential Transfer Learning. Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 213–227.

Liang C, Lung H, Li C. 2014. Overview of Grammatical Error Diagnosis for Learning Chinese as a Foreign Language. Proceedings of the 22nd International Conference on Computers in Education.

Lung H, Liang C, Li C. 2015. Overview of the NLP-TEA 2015 Shared Task for Chinese Grammatical Error Diagnosis. Proceedings of The 2nd Workshop on Natural Language Processing Techniques for Educational Applications, pages 1–6.

Lung H, Liang C, Li C et al. 2016. Overview of NLP-TEA 2016 Shared Task for Chinese Grammatical Error Diagnosis. Proceedings of the 3rd Workshop on Natural Language Processing Techniques for Educational Applications, pages 40–48.

Gaoqi R, Qi G, Baolin Z and Endong X. 2017. IJCNLP-2017 Task1: Chinese Grammatical Error Diagnosis. Proceedings of the 8th International Joint Conference on Natural Language Processing., Shared Tasks, pages 1-9.

Chen L, Junpei Z, et al. 2018. A Hybrid System for Chinese Grammatical Error Diagnosis and Correction. Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications, pages 60–69.

Yuanyuan Z, Nan J, WeiWei S, et al. 2018. Overview of the NLPCC 2018 Shared Task: Grammatical Error Correction. In NLPCC 2018, LNAI 11109, pp. 439–445.

Kai F, Jin H, and Yitao D. 2018. Youdao's Winning Solution to the NLPCC-2018 Task 2 Challenge: A Neural Machine Translation Approach to Chinese Grammatical Error Correction. In NLPCC 2018, LNAI 11108, pp. 341–350.

Endong X. 2018. "HSK dynamic composition corpus.", http://hsk.blcu.edu.cn/.

Devlin J, Chang M, Lee K, et al. 2018. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. In arixv: 1810.04805.

Yinhan L, Myle O, Naman G, et al. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. In arxiv: 1907.11692.

Yiming C, Wanxiang C, Ting L, et al. Pre-Training with Whole Word Masking for Chinese BERT. In arxiv: 1906.08101.

Named-Entity Based Sentiment Analysis of Nepali News Media Texts

Birat Bade Shrestha, Bal Krishna Bal
Information and Language Processing Research Lab,
Department of Computer Science & Engineering,
Kathmandu University, Dhulikhel, Kavre, Nepal
badebirat@gmail.com, bal@ku.edu.np

Abstract

Due to the general availability, relative abundance and wide diversity of opinions, news Media texts are very good sources for sentiment analysis. However, the major challenge with such texts is the difficulty in aligning the expressed opinions to the concerned political leaders as this entails a non-trivial task of named-entity recognition and anaphora resolution. In this work, our primary focus is on developing a Natural Language Processing (NLP) pipeline involving a robust Named-Entity Recognition followed by Anaphora Resolution and then after alignment of the recognized and resolved named-entities, in this case, political leaders to the correct class of opinions as expressed in the texts. We visualize the popularity of the politicians via the time series graph of positive and negative sentiments as an outcome of the pipeline. We have achieved the performance metrics of the individual components of the pipeline as follows: Part of speech tagging – 93.06% (F1-score), Named-Entity Recognition – 86% (F1-score), Anaphora Resolution – 87.45% (Accuracy), Sentiment Analysis – 80.2% (F1-score).

1 Introduction

In recent times, the way the general public acquires news has drastically changed. Traditional news sources such as television, radio and printed newspapers are in a steady decline in terms of use and consumption. Nowadays, most of the people, especially those from the younger generations read the news on the web either from social media or online news portals. As the internet has become more accessible and available to the general public, we have seen a rapid growth in the number of online news portals and blog sites. Almost all of the established media houses have gone online thereby maintaining online news portals besides the hard copy versions. This makes news media texts a very good resource for sentiment analysis in the socio-political domain.

In most countries the popularity of the politicians (especially, the head of state) is tracked by the media houses, independent organizations as well as the party the politicians are affiliated to. The approval ratings show how popular or unpopular a politician is in the view of the general public. Politicians make change to their policies as well as their public persona so that their approval ratings can improve. Positive approval rating can even suggest the likelihood of a politician wining an election. Approval ratings are generally calculated by conducting opinion polls in a particular sample population. Another way of calculating the approval rating can be by finding out what kind of views are being expressed about a politician in printed news media articles. In the context of our country, such approval ratings are not available. This work presents a way of calculating popularity ratings and thus helping to determine how popular or unpopular a Nepali politician is by analyzing the news media texts.

The task of analyzing sentiments in the news media texts has its own set of challenges. A news article may contain sentiments or opinions expressed over more than one politician. Worse, even a sentence might refer to sentiments expressed over multiple politicians. From this perspective, the task demands that the named-entity or the political leader being referred to is accurately identified and resolved in terms of the pronominal references used in the text before moving to the task of aligning the corresponding sentiment to the named-entity. Once the two phase task of named-entity resolution and sentiment alignment is complete, we present the results in the form of a time-series popularity or trending graph. Such a representation would be of interest to a wide range of target audience – the political

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 114–120
Suzhou, China, December 4, 2020.

leader himself/herself, his/her political affiliation, the general public and media houses.

Automating the task is not simple or trivial as the whole process is quite technically involved and requires a series of NLP sub-tasks to be accomplished before we finally reach the end of the pipeline. We describe the pipeline in Section 3 of this paper.

In terms of accomplishing the work, our major contributions can be listed as follows:

a) Developed a Part-of-Speech (PoS) tagger for Nepali
b) Developed a Named-Entity Recognition (NER) classifier for Nepali
c) Developed a rule-based Anaphora Resolution module for Nepali
d) Manually labelled a sentence level Sentiment Corpus of 3490 sentences from Nepali News Media texts
e) Developed a Machine Learning based Sentiment Classifier based on the Sentiment Corpus

2 Related Work

There have been a few research works on Sentiment Analysis in Nepali texts. In one of the first works on Sentiment Analysis for the Nepali Language, Gupta and Bal developed a lexical resource namely the Bhavanakos (Gupta & Bal, 2015).

Yadav & Pant (2014) used a machine learning approach to determine if movie reviews were positive or negative. Their architecture consisted of 3 major components, viz., Pre-Processing, Feature Extraction and Classification. In the Pre-Processing phase, they performed the steps such as whitespace and special character removal, abbreviations expansion, stemming, stop word removal, negation handling, PoS tagging, named-entity recognition etc. To train the model they extracted features such as TF-IDF, positive word count, negative word count, presence of polar words etc. Using these parameters they were able to train a Naïve Bayes classifier and reported a precision of 79.23%, a recall of 78.57% and F-score of 78.90%. Their data set consisted of 500 samples: 250 positive and 250 negative (Yadav & Pant, 2014).

In an undertaking similar to sentiment analysis, Shahi & Pant (2018) used different text mining techniques to address the text classification problem for Nepali news media text. The researchers compared the accuracy of three machine algorithms: Naïve Bayes, Neural Network and Support Vector Machine to classify text according to their content. Their architecture consisted of 3 major components: Pre-Processing, Feature Extraction and Machine Learning. In the Pre-Processing phase they performed the steps such as tokenization, special symbol and number removal, stop word removal and word stemming. To extract the feature from the dataset they used TF-IDF. The researchers have used two instances of the SVM: SVM with Linear Kernel and SVM Radial Basis Function kernels. SVM is a binary classifier but the problem of text classification is a multi-class problem. In order to mitigate this issue the researchers adopted a one-vs.-rest approach. The Neural Network they used was a simple dense backpropagation multilayered perceptron with stochastic gradient descent optimization. The researchers used a five-fold validation method. Their result showed that SVM with RBF kernel outperformed the other three algorithms with an average accuracy of 74.65%. Linear SVM had an average accuracy of 74.62%, Multilayer Perceptron Neural Networks had 72.99% and lastly the Naive Bayes had an accuracy of 68.31% (Shahi & Pant, 2018).

Researchers have used sentiment analysis techniques to extract opinion from News Media texts as well. Thapa & Bal (2016) compared the accuracy of Machine Learning algorithms to classify sentiments expressed in Nepali news media text. They tested three algorithms: Support Vector Machine, Multinomial Naive Bayes and Logistic Regression using a 5-fold cross validation method. Their dataset consisted of 384 book and movie reviews. In the dataset, 179 were positive and 205 were negative sentences. To extract the feature from the dataset, they used four methods: Bag-of-Words, Bag-of-Words (with stopwords removed), TF-IDF and TF-IDF (with stopwords removed). The results obtained from their experiments showed that the F1-score of Multinomial Naive Bayes Algorithm was higher when taking TF-IDF (with stopwords) (Thapa & Bal, 2016).

In addition to this, Kafle (2019) implemented a sentiment based popularity tracker for Nepali politicians. In his research, he used Nepali news media text published in English as his data source. He tracked the popularity of Nepali politicians based on two parameters: growing popularity,

diminishing popularity. To track the popularity of a particular politician, he carried out a sentence level sentiment analysis and assigned sentiment scores to each article with respect to that politician. His architecture can be divided into three main phases. The first phase was Named-Entity Extraction where all the named-entities in the articles were extracted. In the second phase, pronomial anaphora were resolved by replacing the pronouns with the named-entities they were referring to. And in the third and final phase sentiments were extracted from the articles tokenized into sentences. To extract the sentiments from the tokens they used a lexicon and rule-based sentiment analysis tool called Valence Aware Dictionary and sEntiment Reasoner (VADER).

Nepali language is a morphologically rich and complex language. In order to mine opinions from Nepali texts, the text classifier being used should be able to incorporate specific language features before classifying the text (Shahi & Pant, 2018). Different techniques have been used in order to extract sentiment from Nepali text. Researchers have employed learning based as well as rule and lexicon based approach for sentiment classification. To extract features from Nepali text, techniques like TF-IDF, Bags-of-Words etc are used. But the problems with these techniques are that they do not consider the context of the words. It has also been observed that removing stop words has no significant effect on the evaluation metrics and the performance of the classifier (Thapa & Bal, 2016).

3 Methodology

We propose the following framework to address the given research problem, which consists of a pipeline of six components:

1. Data Collection
2. Pre-Processing
3. Parts-of-Speech Tagging
4. Named-Entity Recognition
5. Anaphora Resolution
6. Sentiment Analysis

3.1 Data Collection

A multi-threaded scraping framework was developed in order to facilitate the data collection process. The data was gathered by scrapping news articles from four online news portals. The scrapping framework scraped the news articles from online news portals and saved them in a data repository. The news portals were chosen based upon their influence as a mainstream news source in the Nepali news media. The online portals from which the articles were scrapped are:

i. Kantipur Daily [1]
ii. NagarikNews [2]
iii. Online Khabar [3]
iv. Setopati [4]

3.2 Pre-Processing

The Pre-Processing component consists of two sub-components: Article Cleaning and Article Lemmatizing. First of all, badly encoded characters are removed from the articles. The scrapping framework itself is equipped with the functionality to strip the unnecessary HTML tags and JavaScript codes. Unnecessary punctuation symbols are also removed from the articles and the articles are prepared for the next phase by tokenizing them.

Secondly, the Article Lemmatizing sub-component takes care of the removal of certain suffixes at the end of each tokenized word such as **लाई, मा, हरू, को** etc. Need to note that most inflections in the Nepali language are caused by the postpositions that are placed after nouns, verbs etc. The list of post-positions was obtained from sanjaalcorps [5] .The Article Lemmatizing sub-component splits the root word and the suffixes but does not remove the suffix altogether as it is required in the subsequent phase.

3.3 Part-of-Speech Tagging

The Part-of-Speech (PoS) tagging component assigns lexical category to each word in a text. A Part of Speech is a category of words that have similar grammatical properties. The most common PoS for English language are noun, verb, adjective, adverb, pronoun, preposition, conjunction, interjection, numeral, article or determiner.

[1] https://ekantipur.com/
[2] https://nagariknews.nagariknetwork.com/
[3] https://www.onlinekhabar.com/
[4] https://www.setopati.com/
[5] https://github.com/sanjaalcorps/

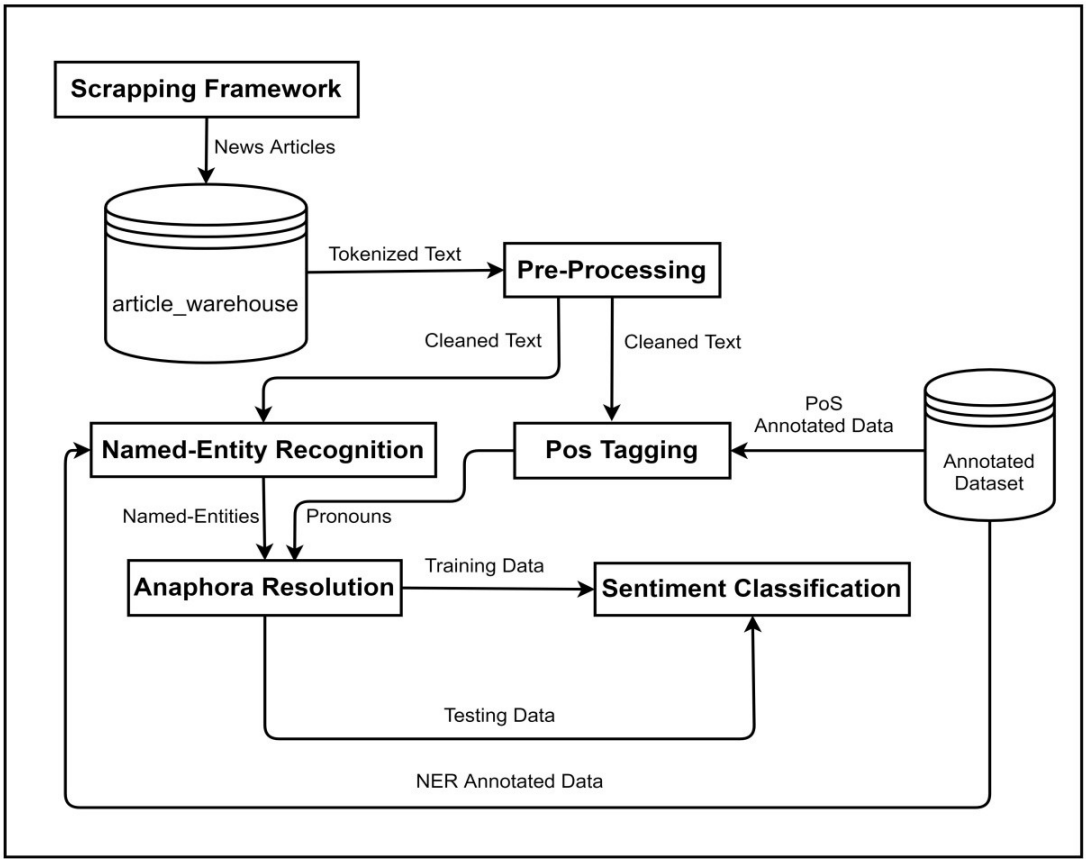


Figure 1: System Architecture

We trained the statistical based Trigrams 'n" Tagger (TnT) tagger with the PoS tagged corpus of Nepali available at the official website of the Center for Language Engineering [6]. The TnT tagger is based on the work of Thorsten Brants (Brants, 2002). If the tagger encounters a new word that has not been trained before, it will tag that word as 'Unk' or Unknown. In order to address the 'Unk' part of speech category, we added a few articles from our corpus into the training dataset.

A 10-fold validation method was used to validate the model. The results showed that the trained model had an average accuracy of 90.05%, precision of 96.55%, recall of 91.3% and F1-score of 93.06. It was also found that lemmatizing the text increased the accuracy of the model by 10%.

3.4 Named-Entity Recognition

A Named-Entity refers to a real-world object such as the name of the person, organization, location etc. In Named-Entity Recognition, such named entities are identified and tagged in text. A Named-Entity Recognition classifier reads the texts, identifies the named entities and classifies them accordingly. For the NER component, we train the StandardNERTagger. This NER classifier is based on the work of Finkel et al (Finkel et al., 2005). It uses an advanced statistical learning algorithm and therefore is relatively computationally expensive. For this work, we used the dataset made available by (Singh et al., 2019). The dataset follows the standard CoNLL-2003 IO format (Sang and Meulder, 2003). The dataset consists of two tab separated fields: the word, named entity tag with one word per line. In order to enhance the performance of the tagger, we extended the dataset with data from our own corpus. We again used a 10-fold validation method to validate the model. The model had an average F1-score of 86%. We present the Precision, Recall and F1-scores for the named-entity tags in Table no 1.

[6] http://www.cle.org.pk/

Tag	Precision	Recall	F1-score
PER	97	85	90
LOC	89	66	76
ORG	87	74	80
O	97	99	98

Table 1: Performance metrics of NER

3.5 Anaphora Resolution

Anaphora resolution refers to the task of correctly resolving the pronominal references in terms of the referred Named-Entity in texts. The referring word is called anaphora and the referenced word is called antecedent.

We implemented a rule-based method based on the Lappin and Leass algorithm to resolve the anaphora on Nepali news media text. The algorithm uses a simple weighting scheme that balances the effects of recency and sentence structure. The algorithm works by adding a discourse variable for each new entity mentioned in the discourse. The algorithm calculates the degree of salience for each entity by summing the weights from a table of salience factors (Lappin and Leass, 1994).

This component uses the pronouns and other parts of speech tagged by the PoS tagging component and the Named-Entities tagged by Named-Entity Recognition Component. In this research work, we resolve the personal pronouns such as **उनी, उन, उहाँ, उहा, उ, ऊ, वहाँ** only. For testing, the first five sentences of 292 news articles were used thus making it a total of 1460 sentences. Out of the 1460, 600 sentences had no named entities whereas 589 had named entities in them and 271 sentences had pronouns. Out of the 271 pronouns, 237 were resolved correctly and 34 were resolved incorrectly. The accuracy of the algorithm was found to be 87.45 %.

3.6 Sentiment Analysis

In this section, we discuss how the sentiment analysis model was developed.

3.6.1 Dataset

Unfortunately, a publicly available sentence level sentiment annotated dataset for Nepali language is not available. Most previous research deals with document level sentiment analysis so they are of very little use for this work. The only option that remained was to manually label the dataset. A total of 3490 sentences were labeled manually. The sentences were classified into one of the following two classes; Positive and Negative. Out of these 3490 sentences, 2676 were positive sentences and 814 were negative. To make the dataset balanced, we included an equal number of positive (814) and negative (814) sentences in the training dataset.

3.6.2 Word Embedding

In Natural Language Processing, representing words in multi-dimensional vector space can improve the performance of a learning based algorithm (Mikolov et al., 2013). In this work, we have used Word2Vec (Mikolov et al., 2013) and FastText (Bojanowski et al., 2016) to embed the words primarily for the purpose of feature extraction for the next component in the pipeline, i.e., Sentiment Classification. There exist pre-trained embedding models for the Nepali language texts (Lamsal, 2019). However, to ensure better coverage, we went for embedding models based on text corpus developed for this research work. The trained model represents each word from the corpus in a 300 dimensional vector space thereby making it 19339 unique words. Four models, Word2Vec with CBOW, Word2Vec with skipgram, FastText with CBOW and FastText with skipgram were trained for testing purposes. To extract the necessary feature vector, firstly the words in the sentences were embedded, and then we averaged the embedding vector of each word in a given sentence. Finally this feature vector was used for sentiment classification. ,

From the initial experimentations, we found out that the classifiers were performing better when the words were embedded using the skipgram parameter. The results obtained are presented in Table no 2.

Model	Parameter	F1
Word2Vec	Continuous Bag of Words	70
	Skip Gram	**80.2**
FastText	Continuous Bag of Words	66.4
	Skip Gram	**78.7**

Table 2: Performance Evaluation of Embedding Methods

3.6.3 Sentiment Classification

For classifying the sentiments in this work, the initial plan was to use Recurrent Neural Network

specialized for Natural Language Processing but due to the small size of our dataset, we used three other machine learning algorithms: Support Vector Machine, Decision Tree and Random Forest.

4 Results

In order to test the performance of the sentiment classifiers, 10-fold cross validation method was used. From the experiments conducted, we found that Support Vector Machine (SVM) with Word2vec embedding (skipgram) had the overall highest performance metrics with an accuracy of 80.15%, precision of 80.4%, recall of 80.2% and F1-score of 80.2%. All the averaged results obtained from the experiments are presented in Table no 3.

Algorithm	**Embedding**	**F1-score**
SVM	**Word2Vec**	**80.2**
	FastText	78.7
Random Forrest	Word2Vec	77.1
	FastText	76.6
Decision Tree	Word2Vec	68.2
	FastText	67.9

Table 3: Performance evaluation of SVM, Random Forrest and Decision Tree

For visualization, popularity trends of three most prominent Nepali politicians from 2018/04 to 2018/09 were plotted. The popularity graphs are shown in Figure no 2.

5 Conclusion

We implemented a named-entity based sentiment analysis framework in this research work in order to distill the outlook expressed towards the politicians in the news media. Initially, the names of the politicians in the news articles were identified and the pronominal expressions that were referring to the names were resolved. Sentences with the name of the politicians were then extracted and classified according to the sentiment expressed towards them.

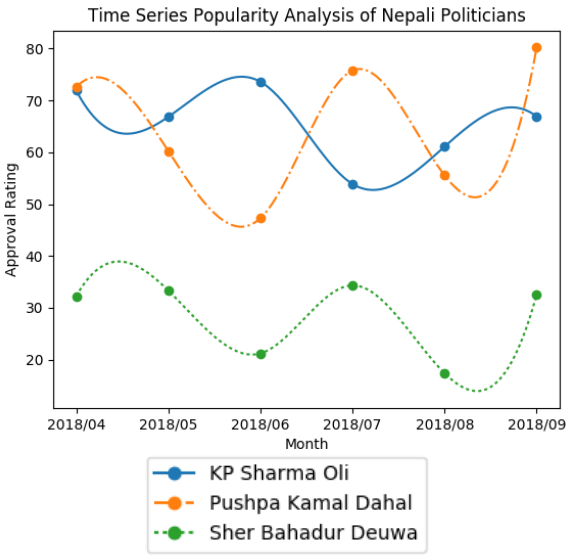


Figure 2: Time Series Popularity Analysis of Nepali Politicians

Support Vector Machine had the overall highest performance metrics for classifying the sentiments expressed in the sentences. We experimented with different embedding techniques and found that Word2Vec with skipgram was the optimal option for feature extraction. This combination of classifier and embedding technique was used to classify the sentences in the articles. Finally, as an application of our research work, we presented the results in the form of a time-series popularity graph or a trending graph.

For each component of the pipeline, we were able to achieve a relatively higher performance metric. Nevertheless, there are some limitations to our work. The components of the proposed framework are based on probabilistic and rule based models, which underperform compared to the neural network models. We could not go for the latter because our dataset is not large enough. Similarly, the anaphora resolution component only resolved pronouns. Other expressions referring to an entity were ignored all together. Furthermore, we have not dealt with the opinion holder or the target explicitly in terms of opinions in this work although named-entities can also be related to the target in many aspects.

The performance of the overall framework can be further enhanced by using more advanced variants of Neural Networks, which are specialized for Natural Language Processing tasks. Different variants of RNN have been used

for Parts-of-Speech tagging as well as Named-Entity Extraction with considerable accuracy and success for other languages. These Neural Networks can be used for sentiment classification as well, given we have sufficient data. So, one aspect of future enhancement to the work would definitely be increasing the dataset. There are other areas of improvements to this work. For anaphora resolution, Graph Based Neural Networks seem to be better as graph based neural networks are more effective in handling non-linear data. Furthermore, latest embedding techniques such as BERT, ELMO, etc. can be used to more accurately embed the context of words within a sentence and thus get better results.

References

Abhimanu Yadav and Ashok K. Pant. 2014. Sentiment Analysis on Nepali Movie Reviews using Machine Learning. *Journal for Research and Development.*

Ashok K. Chandra, Dexter C. Kozen, and Larry J. Stockmeyer. 1981. Alternation. *Journal of the Association for Computing Machinery*, 28(1):114-133. https://doi.org/10.1145/322234.32224.

Chandan Gupta and Bal Krishna Bal. 2015. Detecting Sentiment in Nepali texts: A bootstrap approach for Sentiment Analysis of texts in the Nepali language. 1-4. 10.1109/CCIP.2015.7100739.

Erik F. Tjong Kim Sang and Fien De Meulder. 2003. Introduction to the conll-2003 shared task: Language-independent named entity recognition. *In Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003 - Volume 4*, CONLL '03, pages 142–147, Stroudsburg, PA, USA. Association for Computational Linguistics.

Jenny Rose Finkel, Trond Grenager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. *Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005)*, pp. 363-370.

Kamal Kafle. 2019. Popularity Tracking and Trend Analyses of Political Figures Based on Online News Data (Master's thesis). Kathmandu University

Lal Bahadur Reshmi Thapa and Bal Krishna Bal. 2016. Classifying sentiments in Nepali subjective texts. 1-6. 10.1109/IISA.2016.7785374.

Oyesh Mann Singh, Ankur Padia, and Anupam Joshi. 2019. Named Entity Recognition for Nepali Language.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2016. Enriching word vectors with subword information. *arXiv preprintarXiv:1607.04606*.

Rabindra Lamsal. 2019. 300-Dimensional Word Embeddings for Nepali Language. IEEE Dataport.http://dx.doi.org/10.21227/dz6s-my90

Shalom Lappin, Herbert J. Leass. 1994. An algorithm for pronomial anaphora resolution. Computational Linguistics 20, 535–561

Tej Bahadur Shahi and Ashok Kumar Pant. 2018. Nepali news classification using Naïve Bayes, Support Vector Machines and Neural Networks. 1-5. 10.1109/ICCICT.2018.8325883.

Thorsten Brants. 2002. TnT: A Statistical Part-of-Speech Tagger. ANLP. 10.3115/974147.974178.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Distributed representations of words and phrases and their compositionality. *CoRR*, abs/1310.4546.

SEMA: Text Simplification Evaluation through Semantic Alignment

Xuan Zhang[1,2], Huizhou Zhao[1,*], Kexin Zhang[1], and Yiyang Zhang[3]

[1]School of Information Science,
Beijing Language and Culture University
Email: xk18zx@126.com, zhaohuizhou@blcu.edu.cn, jacquelinelopze@outlook.com
[2]Department of Automation,
Tsinghua University
[3]College of Chinese Studies,
Beijing Language and Culture University
Email: bournezyy@163.com

Abstract

Text simplification is an important branch of natural language processing. At present, methods used to evaluate the semantic retention of text simplification are mostly based on string matching. We propose the SEMA (text Simplification Evaluation Measure through Semantic Alignment), which is based on semantic alignment. Semantic alignments include complete alignment, partial alignment and hyponymy alignment. Our experiments show that the evaluation results of SEMA have a high consistency with human evaluation for the simplified corpus of Chinese and English news texts.

1 Introduction

Text simplification is a rewriting operation that aims to improve the comprehensibility of the text by modifying, deleting, simplifying human-readable text. It tried to retain the core semantics of original text while improving readability of the text. In natural language processing tasks, long and complex sentences will bring about various problems, for example, the quality of grammatical analysis depends on the length and the grammar difficulty of texts directly, and complex sentences may cause ambiguity during machine translation(Chandrasekar and Srinivas, 1997). Therefore, text simplification is often used in the preprocessing steps of other NLP tasks. In addition, text simplification is also used to rewrite reading materials for children, second language learners, readers with aphasia and other people with low reading comprehension skills(Carroll J, 1998). As related researches are in the early stage, the results of text simplification cannot meet the needs of the audience well. One of the difficulties is the lack of reasonable text simplification evaluation indicators. At present, most evaluation methods are conducted by experts or machine translation evaluation indicators. Therefore, researches on how to analyze the results of text simplification has important application value.

Text simplification mainly includes vocabulary and semantic structure simplification. The main operation is text segmentation, that is, rewriting a single sentence into one or more simpler sentences while preserving the main semantics(Sulem et al., 2018b). Text simplification has gradually attracted attention in recent years(Xu et al., 2016; Saggion and Horacio, 2017; Saggion et al., 2012), it should be evaluated from three aspects: fluency (ie: grammatical correctness), correctness (ie: semantic retention) and simplicity (ie: degree of text simplification). Initially, experts can only evaluated the results through three aspects, and the final score is based on the Likert scale[1]; Later, someone proposed to use readability indicators to evaluate text simplification, but because the readability indicators are designed for passage-level texts, the application effects at the sentence level are not very prominent(Coster and Kauchak, 2011). In recent years, the evaluation indicators of machine translation have been increasingly used in the evaluation of text simplification, including BLEU, ROUGE based on N-gram and WER, TER based on edit distance.

In machine translation tasks, BLEU is the most widely used evaluation indicators, which was proposed in 2002. The original purpose is to replace the manual evaluation of translation results. The quality of the machine translation task is mainly evaluated by evaluating the difference between the

*Corresponding author: zhaohuizhou@blcu.edu.cn

[1]Likert scale is one of the most commonly used scoring aggregate scales. It was developed by American social psychologist Likert in 1932 on the basis of the original aggregate scale. The scale consists of a set of statements. Each statement has five answers: "strongly agree", "agree", "not necessary", "disagree" and "strongly disagree", which are recorded as 5, 4, 3, 2, 1, and the final score is the sum of score for each aspect.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 121–128
Suzhou, China, December 4, 2020.

output generated by model and the reference. It has low computational cost and is highly correlated with human evaluation, so it is widely used. Elior Sulem's experiments show that Since the main operation of text simplification is text segmentation, involving semantic structure splitting, BLEU did not show a high degree of relevance to manual evaluation in terms of grammar and semantic retention of 70 pairs of sentences (Sulem et al., 2018c). In addition, in terms of simplicity assessment, BLEU shows a negative result which penalized simplified sentences highly.

SARI is an evaluation indicator based on reference sentences proposed in 2016(Xu et al., 2016). It focuses on the aspect of words added, deleted, and retained, but it cannot evaluate sentences at semantic level. SAMSA is a semantic structure-based evaluation indicator proposed in 2018(Sulem et al., 2018a), but it relies too much on string matching in the judgment of semantic consistency, which leads to low semantic retention calculation results for simplified text. Based on the characteristics of these evaluation indicators, this research proposes a text simplification evaluation indicator SEMA based on semantic alignment.

The contribution of this paper is to propose a semantic retention evaluation indicator of text simplification based on semantic alignment. Semantic alignment includes **complete alignment**, **partial alignment** and **hyponymy alignment**. Different semantic alignment weights are given according to the degree of semantic alignment, so as to reasonably evaluate the semantic retention of text simplification of different rewriting methods.

2 Related Work

2.1 Universal Cognitive Conceptual Annotation(UCCA)

The current traditional syntactic structure cannot directly reflect the semantic difference of the text, for example:"John took a shower." (a) and "John showered." (b) Syntactic analysis will regard them as different structures, but at the semantic level, (a) and (b) are similar. The UCCA (Universal Conceptual Cognitive Annotation)(Abend and Rappoport, 2013) proposed in 2013 avoids this defect. Its scene-based semantic structure annotation method aims to extract the scene graph formed by main relation and participants to represent the main semantic infomation in the text.

The scenes of UCCA represent motions, actions or states that persist in time, and are divided into State (S) and Process (P). A State represents a continuous state in time, such as:"There has been conflict in Syria for the last nine years." A Process describes an event that is evolving and unfolding in time, such as: "The dog runs into the house." Each scene contains a main relation, one or more participants (including location information), such as:"John kicked his ball." In this scene, the participants are "John" and "his ball", the relation is "Kicked".

The UCCA structure is a directed acyclic graph, and the smallest meaningful unit is on the leaf node (that is, the word in the text). For units that cannot form a scene, the UCCA sets a category Centers (C) to represent the subunits of a non-scene unit, and there may be one or more C in a non-scene unit. Modifiers (including qualifiers) are marked as Elaborator (E). For example, in the non-scene unit "his ball", "his" is E, and "ball" is C.

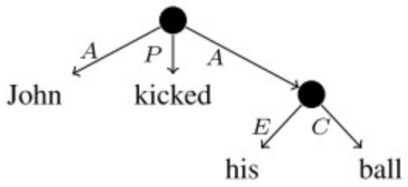


Figure 1: The result of "John kicked his ball" by UCCA

In actual contexts, more complicated situations often occure: one scene may be a participant of another scene. For example, in the sentence "The report says that the USA can be war criminals", "the USA can be war criminals" is A in the scene where "says" is a relation; one scene can also be E in another scene, such as the sentence: "The day Tom arrived in Beijing was Friday", the scene "Tom arrived in Beijing" is E that modifies "The day", and "The day" is A of the scene "The day was Friday".

UCCA is a semantic annotation method as opposed to syntactic analysis. It is portable between various fields and languages, and is not sensitive to semantic-retain grammatical changes. In addition, it can accommodate more semantic differences. In this research, the TUPA tool is used to obtain the UCCA annotation result (Hershcovich et al., 2017), it uses the NN classifier and BiLSTM model for training, inputting text and outputting UCCA result.

2.2 Simplification Automatic evaluation Measure through Semantic Annotation(SAMSA)

SAMSA is the first indicator to evaluate the quality of Text Simplification (TS) system at the semantic structure level. It uses UCCA based on the concept of scene to try to reasonably evaluate the text simplification results in terms of semantic rather than syntax(Sulem et al., 2018a). SAMSA extracts the scene of the input sentence, and after identifying the relation and participants, it does the word comparision calculation with output sentence It believes that the result of a high-quality text simplification should be: each input scene is mapped to the output sentence one by one, the smallest unit of the relation and the participants (see later) can be matched in the output sentence. SAMSA is a non-referenced automatic evaluation method. Elior Sulem's experiments show that SAMSA has a high relevance to human evaluation in terms of semantic retention. SAMSA is explained in detail below.

SAMSA is based on two external tools-UCCA and Word Alignment. UCCA decomposes each input sentence S into a set of scenes$\{\mathrm{SC}_1, \mathrm{SC}_2, \cdots \mathrm{SC}_\mathrm{n}\}$, each scene$\mathrm{SC}_i$ contains one main relationMR_i and one or more participantsA_i; Word Alignment aligns the words of the input sentence with one or zero words of the output sentence to form a set A, which can identify synonym substitution (start/begin) and stemming (run/ran). n_{inp} is the number of scenes of input, $\mathrm{n}_{out}$is the number of sentences of output (S_1, S_2 ,.., S_{nout}). Firstly, SAMSA aligns the input scene and the output sentence. There are two cases:

1.$\mathrm{n}_{inp} \geq n_{out}$: in this case, we compute the maximal Many-to-1 correspondence between Scenes and sentences. To align each input scene with the output sentence, SAMSA gets the number of word matches between each scene and each output sentence according to the word alignment A, and select the sentence with the highest matching degree to align. If $\mathrm{n}_{inp} = n_{out}$, once a sentence is matched to a scene, it cannot be matched to another one.

$$M*(SC_i) = argmax_s score\,(SC_i, S) \quad (1)$$

2.$\mathrm{n}_{inp} < n_{out}$: In this case, a scene will necessarily be split across several sentences. As this is an undesired result, SAMSA assigns this instance a score of zero.

For the scenes of input$\{\mathrm{SC}_1, \cdots \mathrm{SC}_{\mathrm{ninp}}\}$, the sentences of output$\{\mathrm{S}_1, \cdots, \mathrm{S}_{\mathrm{nout}}\}$ and their mapping relationship$M^*\,(SC_i)$, the calculation formula of SAMSA is as follows:

$$SAMSA = \begin{cases} \frac{n_{out}}{n_{inp}} \frac{1}{2n_{inp}} \sum\limits_{SC_i} \left[II_{M*(SC_i)}\,(MR_i) + \frac{1}{k_i} \sum\limits_{j=1}^{k_i} II_{M*(SC_i)}\left(Par_i^{(j)}\right) \right], & n_{inp} \geq n_{out} \\ 0, & n_{inp} < n_{out} \end{cases} \quad (2)$$

$\mathrm{MR}_i$is the smallest unit of relation in SC_i, $\mathrm{Par}_i^{(j)}\ (j = 1, \cdots, k_i)$ is the smallest unit of participants in SC_i. The smallest unit is the child node marked as C in the UCCA graph starting recurrence from P/S and A until the leaf node. If the participant is a scene, its smallest unit is the main relation of the scene. For example, the center of "the tallest building in the world" (u1) is "the tallest building". The center of the latter is "building", which is a leaf node. Therefore, the smallest unit of u1 is "building".

$\mathrm{II}_s\,(u)$ defines a function with a value between 0 and 1. If there is a word alignment in u and s, the value is 1, otherwise the value is 0. SAMSA sets a penalty factor $\mathrm{n}_{out}/\mathrm{n}_{inp}$ to penalize the case of $\mathrm{n}_{inp} > n_{out}$. In addition, SAMSA-abl is also set as the calculation indicator for removing the penalty coefficient, and the calculation is shown in formula 3. Elior Sulem's experiment (Sulem et al., 2018a) shows that the evaluation result of the SAMSA-abl indicator (0.54), which removes the penalty coefficient, is better than SAMSA. It indicates that the penalty coefficient will over-punish the situation of $\mathrm{n}_{inp} > n_{out}$, so this research improves the indicator based on SAMSA-abl.

$$SAMSA = \begin{cases} \frac{1}{2n_{inp}} \sum\limits_{SC_i} \left[II_{M*(SC_i)}\,(MR_i) + \frac{1}{k_i} \sum\limits_{j=1}^{k_i} II_{M*(SC_i)}\left(Par_i^{(j)}\right) \right], & n_{inp} \geq n_{out} \\ 0, & n_{inp} < n_{out} \end{cases} \quad (3)$$

To make the calculation process of SAMSA-abl clearer, we take the input sentence (a) "About 13 million Syrians had to leave their homes because of danger." and the simplified sentence (b) "About 13 million had to leave their homes." as an example. The smallest unit of the main relation of input scene is "leave", and the smallest unit of participants is "About, 13, million", "Syrians" and "homes". In all the smallest units, only "Syrians" in the simplified sentence fails to match the input sentence. Therefore, $\mathrm{II}_{M*(sc_1)}\,(MR_1)$ is 1, $\mathrm{II}_{M*(sc_1)}\,(Par_i)$ is 1+0+1=2(k=3), and the score of (b) is 1/2* （1+1/3*2） =0.83.

3 Text Simplification Evaluation Through Semantic Alignment(SEMA)

SEMA is a further optimization of the SAMSA indicator, including two parts: 1. The basic for-

mula SEMA-base (basic formula) is obtained by calculation when $\mathrm{n}_{inp} < n_{out}$ is added on the basis of SAMSA-abl; 2. In terms of indicator calculation strategy, semantic alignment is used to replace the string alignment and it mainly includes three semantic alignment methods: full alignment (SEMA-base), partial alignment, and hyponymy alignment.

3.1 SEMA-base

SAMSA believes that when $\mathrm{n}_{inp} < n_{out}$, a scene is broken into multiple sentences, which destroyes the structure of the scene, so the score is 0. However, in the corpus used in this research, there are more texts that meet $\mathrm{n}_{inp} < n_{out}$. For example, in the original sentence "Central Park Tower has just become the tallest residential building in the world", the simplified text is divided into four sentences:" (1)Central Park Tower is a building in New York. (2)There are only apartments in this building. (3)There are no offices in this building. (4)Now, it is the tallest building with apartments in the world." Although this text divides a scene into multiple sentences, from the perspective of reading comprehension, the simplified sentence is easier to understand and also retains the semantics of original sentence. It is unreasonable to get 0 under the condition of $\mathrm{n}_{inp} < n_{out}$.

Based on this point, on the basis of SAMSA-abl, the definition of SEMA-base is shown in formula 4, where when $\mathrm{n}_{inp} < n_{out}$, the simplified text can still get a score.

$$SEMA-base = \frac{1}{2n_{inp}} \sum_{SC_i} \left[II_{M*(SC_i)}(MR_i) + \frac{1}{k_i} \sum_{j=1}^{k_i} II_{M*(SC_i)}(Par^{(j)}) \right] \quad (4)$$

3.2 Computing Strategy changes

SAMSA relies too much on string-match when aligning the words of the input scene and the output sentence, which leads to low evaluation results easily. SEMA changes the calculation and matching method based on SEMA-base, and emphasizes semantic alignment, including complete alignment, partial alignment and hyponymy alignment. Complete alignment is the original SAMSA string-match strategy.

Partial Alignment: SAMSA requires that the smallest unit of the participant in the scene matches the word of the output sentence. For the case where a participant contains multiple smallest units, SAMSA requires that all smallest units should be matched to get score 1, otherwise it is 0. For example, for the input sentence "I like banana, apple and orange.", the participants are "banana, apple, orange". When the output sentence is "I love apple.", only "apple" is matched in the smallest unit of the participant, but the value is 0 according to the SAMSA matching method. Obviously, this is not friendly to sentences that contain part of the smallest unit. Partial alignment calculates the matching degree of every single smallest unit and SEMA-part is defined as shown in formula 5. On the basis of SEMA-base, the parameter m_q is added to represent the number of smallest units of participants, and $Par_i^{(j)(q)}$ is the qth smallest unit of the jth participant in the ith scene.

$$SEMA-part = \frac{1}{2n_{inp}} \sum_{SC_i} \left[II_{M*(SC_i)}(MR_i) + \frac{1}{k_i} \sum_{j=1}^{k_i} \frac{1}{m_q} \sum_{q=1}^{m_q} II_{M*(SC_i)}\left(Par_i^{(j)(q)}\right) \right] \quad (5)$$

Hyponymy Alignment: In order to summarize the text features of text simplification better and establish a more complete evaluation indicator in terms of semantic evaluation, we observed and disassembled the corpus, compared the manual score with automatic machine score, and found the feature of hyponymy in the corpus. It is a common operation to replace hyponym with hypernym in text simplification. Here, the hyponymy refers to the words with the upper and lower conceptual relationship, and they have a species relationship (Chi, 1989), such as "drinks" is the hypernym of "beer", "fruit" is the hypernym of "kiwi". Generally, the most simplified text has more hyponymy. In this research, based on SEMA-part, we use WordNet's hyponymic relationship network to align the smallest unit of relations and participants which include hyponymy. And it improves the degree alignment between the input scene and the output sentence. Finally, a text simplification evaluation indicator based on semantic alignment SEMA is formed. The calculation formula of SEMA is still shown in formula 5. The difference between SEMA-part and SEMA is only the addition of hyponymy alignment to the semantic alignment. In the end, our experiments proved that SEMA is highly usable in evaluating the semantic retention of Chinese and English text simplification at sentence and passage level. See chapter 4 for more details.

4 Evaluation Experiment Based On Artificial Simplified Corpus

4.1 Corpus

This research uses simplified Chinese and English news corpus for experiments. The simplified

English corpus comes from the English website: News in Levels, which is a free online news website specially designed for English students. Each article is written in three levels, and level 1 is the simplest. Taking level 3 as the benchmark, the semantic retention of level 2 and level 1 is manually judged to be around 70% and 50% respectively. The Chinese news corpus comes from the texts of the Chinese news reading textbook and its corresponding original texts. The texts of the news reading textbook are simplified and adapted for teaching needs. The semantic retention of the adapted text is around 80%. This research collected 200 English passages (three levels), 600 pieces in total, 100 aligned sentences; 100 Chinese aligned sentences.

4.2 English Corpus Experiment

We first perform experiments on SAMSA, SAMSA-abl, SEMA-base, SEMA-part, and SEMA on 100 English sentences. The experimental results are shown in Table 1.

sentence level		
	level1	level2
SAMSA	0.22	0.36
SAMSA-ab1	0.34	0.62
SEMA-base	0.43	0.65
SEMA-part	0.45	0.67
SEMA	0.48	0.69

Table 1: Sentence-level results of SAMSA and SEMA

The results show that SAMSA does not evaluate the semantic retention of each level of corpus very well; after removing the penalty coefficient, SAMSA-abl significantly improves the scores of the two levels. It proves that the penalty coefficient will over-punish the corpus; when considering the case: $\mathrm{n}_{inp} < n_{out}$,the scores of level1 and level2 are improved and the degree of improvement of level1 is more obvious, which also matches the corpus characteristics of level1 (more corpus conforms to $\mathrm{n}_{inp} < n_{out}$); After adding partial alignment and hyponymy alignment, the results of the corpus evaluated by SEMA are closer to the human estimated scores, with level1-score increased to 0.48 and level2-score increased to 0.69. The effect of each optimization strategy on the experimental results is shown in Figure 2.

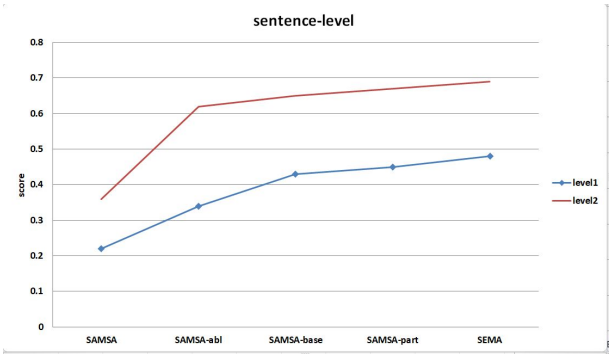


Figure 2: Sentence-level semantic retention evaluation, the improved experimental results of each optimization strategy

SAMSA is proposed to evaluate the sentence-level text simplification system. This research applies it to the passage-level evaluation. Firstly, 35 passages corresponding to 100 alignment sentences are selected for experiment. Based on SEMA-base, the result of level1 is 0.38, and the result of level2 is 0.52.

It can be seen from the experimental results that the overall score of the passage level is lower than the sentence level. This is because in the passage-level evaluation, the length of the passage and the sentence number increase, the scene analysis tool TUPA is unstable. Therefore, it is difficult to extract the scene (that is, multiple sentences extract a large scene), such as the sentence "They based the report on hundreds of interviews and analyses of photos, videos, and satellite images." should be divided into one scene, but in the actual results, "videos, and satellite images" and "Put simply" which is far away are seen as one scene. Since the input scene and the output sentence are aligned according to the maximum number of word matches, and the scene cannot be split clearly, multiple scenes can only be aligned with one sentence. Obviously, it is difficult to find all the semantic information of multiple scenes in one sentence in this case, which directly affects the quality of the indicator evaluation. In order to improve this shortcoming, we splitted the original passages (level3) and then used TUPA for scene analysis. The scene analysis result of each sentence was compared with the simplified whole passage, so the best match can be selected. The final score is averaged.

The experimental results at the passage level are shown in Table 2. "Segmentation+SEMA-base" is an improvement based on SEMA-base. It can be

seen that the division of the passage helps TUPA extract the scene and improve the accuracy of the indicator. In the end, the evaluation results of 35 passages of level1 and level2 increased from the initial 0.24 and 0.26 to 0.53 and 0.69 respectively. When we expand the corpus from 35 passages to 200 passages, level1 and level2 scores are 0.51 and 0.68 respectively, which is consistent with the manual evaluation results.

	passage level	
	level1	level2
SEMA-base	0.38	0.52
Segmentation+SEMA-base	0.44	0.62
Segmentation+SEMA-part	0.45	0.64
Segmentation+SEMA	0.53	0.69

Table 2: Passage-level results of SEMA

As for the passage level, the effect of each indicator optimization strategy on the experimental results is shown in Figure 3. Among them, the improvement of hyponymy alignment is obvious, and the performance on level 1 is particularly prominent. In the final SEMA evaluation results, the passage level score of level 1 is much higher than the sentence level. The main reason is that when we align the sentences, we filter out some improperly aligned sentences, and all sentences at the passgae level participate in the scoring. The scores of these sentences increase the average score at the passage level.

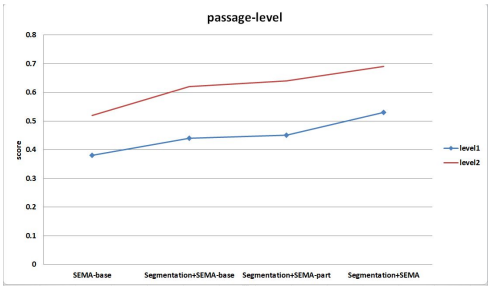


Figure 3: Passage-level semantic retention evaluation, the improved experimental results of each optimization strategy

4.3 Chinese Corpus Experiment

Compared with English, Chinese is consistent with English in the main output sequence of sentences such as subject, predicate and object. For some subsidiary components, such as attributes that modify the subject and object, there are many differences between Chinese and English. The Chinese corpus comes from the adapted Chinese news reading textbook and its corresponding original text. The adaptation methods include but are not limited to: deletion, replacement, and rewriting. According to manual evaluation, the semantic retention of the adapted corpus is about 80% or more.

This research uses semi-automatic processing in the Chinese corpus experiment. There are no tool to analyse Chinese UCCA structure, when extracting the main information of the input sentence, we use Baidu dependency syntax analysis [2] to extract the core word of the sentence (HED) as the main relation, the first-level child nodes of the core words are participants. For example, in the sentence "2004年3月26日全法汉语教学研讨会在巴黎国际大学生城举行。", the relation is "举行", the participants are "研讨会" and "在巴黎". For semantic alignment, we use manual alignment in a non-automated way, and finally conduct experiments based on aligned 100 Chinese sentences. The results are shown in Table 3.

	Adapted text
SEMA	0.804

Table 3: SEMA evaluation results at the Chinese sentence level

Experiments show that in evaluating the semantic retention of the adapted Chinese sentences, SEMA reaches to 0.804, which is consistent with the manual evaluation result. This has great significance for the evaluation of the semantic retention of Chinese text simplification.

5 Conclusion

This research improves the semantic structure based text simplification evaluation measure SAMSA proposed in 2018. There are mainly several aspects: the case of $\mathrm{n}_{inp} < n_{out}$ is considered on the basis of SAMSA-abl; semantic alignment is used to replace string matching, mainly based on three semantic alignments method: Full alignment, partial alignment, hyponymy alignment. Finally, a semantic retention evaluation measure about text simplification SEMA based on semantic alignment

[2]The dependency syntax explains its syntactic structure by analyzing the dependencies of the components in the language unit, claiming that the core verb in the sentence is the central component that dominates other components

is formed. We did experiments on English sentence-level and passage-level. The experimental results show that it is similar to the manual evaluation results, which shows its significance in text simplification evaluation.

When we apply SEMA to Chinese, we summarize the characteristics of Chinese and use dependency syntax analysis to extract the main semantic information in the sentence. Experimental results show that SEMA has high applicability in Chinese corpus, and it is the first semantic retention evaluation indicator based on semantic alignment on Chinese corpus.

In future research, we will continue to use larger corpus to explore SEMA's evaluation methods under different semantic retention thresholds; In addition, the text simplification indicator proposed in this paper only evaluates the semantic retention, other aspects of evaluating texts simplification such as grammaticality and degree of simplification need to be further explored; At the same time, follow-up research should expand the scale of the text corpus and collect multi-subject, multi-genre and multi-length texts to test the usability of our indicator.

Acknowledgments

The research is supported by Science Foundation of Beijing Language and Culture University (supported by "the Fundamental Research Funds for the Central Universities")(19YJ040005); Major Program of National Social Science Foundation of China (18ZDA295); Top-ranking Discipline Team Support Program of Beijing Language and Culture University(JC201902).

References

Omri Abend and Ari Rappoport. 2013. Universal conceptual cognitive annotation (ucca). In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

Xiaohong Yuan Guodong Zhou Bukang Wang, Hongling Wang. 2010. Chinese semantic role tagging based on dependency syntax analysis. *Journal of Chinese Information Processing — J Chin Inf Proc*, 01:25–29. (in Chinese).

Canning Y Devlin S Tait J Carroll J, Minnen G. 1998. Practical simplification of English newspaper text to assist aphasic readers. In *Proceedings of the AAAI-98 Workshop on Integrating Artificial Intelligence and Assistive Technology*, pages 7–10.

R. Chandrasekar and B. Srinivas. 1997. Automatic induction of rules for text simplification. *Knowledge Based Systems*, 10(3):183–190.

Mei Chi. 1989. Talking about hyponymy. *HAN YU XUE XI*, (01):26–28. (in Chinese).

Will Coster and David Kauchak. 2011. Learning to simplify sentences using wikipedia. In *Proceedings of the Workshop on Monolingual Text-To-Text Generation*.

Michael Denkowski and Alon Lavie. 2014. Meteor universal: Language specific translation evaluation for any target language. In *Proceedings of the Ninth Workshop on Statistical Machine Translation*.

Daniel Hershcovich, Omri Abend, and Ari Rappoport. 2017. A transition-based directed acyclic graph parser for ucca. *arXiv preprint arXiv:1704.00552*.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

George Miller. 1995. Wordnet: A lexical database for English. *Communications of the ACM*, 38:39–.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. BLEU: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318.

Saggion and Horacio. 2017. Automatic text simplification. *Synthesis Lectures on Human Language Technologies*, 10(1):1–137.

Horacio Saggion, Elena Gómezmartínez, Esteban Etayo, Alberto Anula, and Lorena Bourg. 2012. Text simplification in simplext. making text more accessible. In *International Conference on Computational Science*.

Matthew Snover, Bonnie Dorr, Richard Schwartz, Linnea Micciulla, and John Makhoul. 2006. A study of translation edit rate with targeted human annotation. In *Proceedings of association for machine translation in the Americas*, volume 200. Cambridge, MA.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018a. Semantic structural evaluation for text simplification. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018b. Simple and effective text simplification using semantic and neural methods. In *Meeting of the Association for Computational Linguistics*.

Elior Sulem, Omri Abend, and Ari Rappoport. 2018c. BLEU is not suitable for the evaluation of text simplification. *arXiv preprint arXiv:1810.05995*.

Wei Xu, Chris Callison-Burch, and Courtney Napoles. 2015. Problems in current text simplification research: New data can help. *Transactions of the Association for Computational Linguistics*, 3:283–297.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Lingus*, 4(4):401–415.

A Corpus Linguistic Perspective on the Appropriateness of Pop Songs for Teaching Chinese as a Second Language

Xiangyu Chi Gaoqi Rao
Beijing Language and Culture University
cxyblcu@163.com， raogaoqi@blcu.edu.cn

Abstract

Language and music are closely related. Regarding the linguistic feature richness, pop songs are probably suitable to be used as extracurricular materials in language teaching. In order to prove this point, this paper presents the Contemporary Chinese Pop Lyrics (CCPL) corpus. Based on that, we investigated and evaluated the appropriateness of pop songs for Teaching Chinese as a Second Language (TCSL) with the assistance of Natural Language Processing methods from the perspective of Chinese character coverage, lexical coverage and the addressed topic similarity. Some suggestions in Chinese teaching with the aid of pop lyrics are provided.

1 Introduction

The appropriateness of pop songs for language teaching has been widely discussed by scholars in recent years. At present, both quantitative and qualitative studies on the songs' application in the English teaching have developed maturely. Meanwhile, quantitative studies have not attracted enough attention in the Chinese teaching. Related studies prefer to explore the songs' potential in teaching assistance in the dimensions of lyrics' linguistic features, teachers' practical experience and students' knowledge. Therefore, this paper focuses on the similarity between lyrics and teaching material for TCSL. The rest of the paper is organized as follows: in section 2 the related work is surveyed, section 3 and 4 describe the CCPL corpus in detail, section 5 and 6 analyze the statistics of character coverage, lexical coverage and topic similarity of the CCPL corpus in TCSL. Conclusions and future works are finally drawn in Section 7.

2 Related Work

In the field of the English teaching, Plitsch (1997) raised the idea of using contemporary lyrics for more inspiring and close-to-reality language teaching. Werner (2012) compared the role of American English and British English in popular songs, and Werner (2018) outlined its didactic potential. Tegge (2017) examined the lexical coverage of pop lyrics in English teaching, using around 1,000 songs in two collections. Applications of Natural Language Processing (NLP) tools could also be found in related work: Penaranda (2006) uses text mining for empirically based genre assignments, involving linguistic anomalies. Napier/Shamir (2018) took a diachronic perspective and quantify emotional changes in lyrics since 1950.

In the field of the Chinese teaching, Shouhui Zhao and Qingsong Luo (1994) were the first to propose the introduction of songs into classes. They believed songs could create a good cultural atmosphere for Chinese as foreign language (CFL) learners, resulting in their high interest in studying. You Fu (2002) argued that songs could help solve the difficulties of traditional listening lessons from the perspective of psychology. Chenxu Yang (2019) affirms that music plays a positive role in enhancing CFL learners' sense of language, strengthening the ability of correcting errors and improving efficiency. Yanjing Wang (2011) puts forward that songs can help CFL learners to correct pronunciation, expand vocabulary, understand rhetoric, recognize proverbs, etc.

Proceedings of the 6th Workshop on Natural Language Processing Techniques for Educational Applications, pages 129–137
Suzhou, China, December 4, 2020.

Based on the above, this paper found that previous studies in TCSL seldom used quantitative methods or NLP tools to extract and analyze language information contained in lyrics. Therefore, we attempt to improve the subordinate role which quantitative searches play in the appropriateness of pop songs for TCSL.

3 Corpus Building

The CCPL corpus contains lyrics in Chinese, even if a minority are in English. Given that this paper focus on the coverage of lyrics' content in TCSL, English words are excluded for noise reduction. Enabling further processing with NLP tools, the original lyrics need to be transferred into annotated contents. Therefore, word segmentation and part of speech (POS) tagging are needed. Words extracted by topic models are also presented for comparison.

3.1 Data Selection

A total of 1,110 pop songs were contained in the CCPL corpus, which were collected from authoritative competitions and lists such as CCTV Spring Festival Gala Evening, Top Chinese Music Awards, etc. The metrics for the selection are as follows: Firstly, pop songs are of high popularity among the mass media and reflect the social changes and the zeigeist. Secondly, the songs in the list tended to be the songs of the year. They were selected by the mass and experts voting among the songs of the year, and the organizer is officially authoritative and has certain credibility.

3.2 Data Annotation

We implemented *jieba*[1], a built-in module in python, to realize the segmentation and POS tagging of the lyrics. Fully automatic annotation of such units produces rather poor results, therefore we corrected the results with the assistant of bi-gram model.

In the process of manual processing, we took The Basic Processing of Contemporary Chinese Corpus at Peking University (PU) and *jieba* POS labeling specification as standard. Several fuzzy phenomena of in word segmentation are briefly introduced as follows.

Numerals and quantities. *Jieba* counts quantitative phrases as a segmentation unit, such as "一个". In this paper, they are divided into numerals and quantities, namely "一/m 个/q".

Separable words. Separable words are regarded as three segmentation units in the PU. This paper chooses to combine them, resulting in expressing concepts more completely and being directly labeled as verbs, such as "舍不得/v".

Overlapping words. The quantitative structure in the form of "ABB" is segmented, namely "一/m 颗/q 颗/q". This is to enable the vocabularies in CCPL to cover as many vocabularies in HSK as possible. Meanwhile, the adjective overlapping form "ABB" and the two-syllable verb and adjective overlapping form "AABB" are not segmented, such as "甜甜蜜蜜/a", "亮堂堂/a".

Morpheme + "们" or word + "们", which indicates the plurals of nouns should be segmented, such as "孩子/n 们/k", while words such as "我们" express a concept independently, we label them as "我们/r".

Monosyllabic verb +directional verb. *Jieba* processes this form into a segmentation unit such as "爱上". We process them into "爱/v 上/vf". This is also intended to cover as many vocabularies in HSK as possible

3.3 Topic Description

Each song in the CCPL corpus was provided with the top ten words extracted by two weighting schemes respectively, namely Textrank and TF-IDF. We took the words as reference to conclude the content of topics. For instance, the topic *Travel* contains words of both the field of *specialty* ("煎饼", "大葱") and the field of *sights* ("泰山", "黄河").The topic *Family* contains words of both the field of *family members* ("老爸", "老妈", "宝贝") as well as the words of *daily life* ("拼命", "风雨无阻", "跑调").

Figure1(a): Topic Family. Figure1(6): Topic Travel.

[1] https://github.com/fxsjy/jieba

Figure2(a): Topic Food.

Figure2(b): Topic Peking Opera.

4 Corpus Overview

The key statistics on the corpus are summarized in this section, such as word length and POS distribution, and its accessibility is also provided.

4.1 Word length

The CCPL corpus comprises 292,609 tokens and 3,320 types of characters together with 208,537 tokens and 12,231 types of words. Figure 3 shows the distribution of word length in detail. Among the 12,231 types of words, disyllabic words occupy the largest ratio, including 8,598 types such as “我们”, which appear 71473 times in total. Monosyllabic words such as “的” come next with 2077 types, which appear 131949 times in total. The longest word type is a six-syllable word, such as “一步一个脚印”.

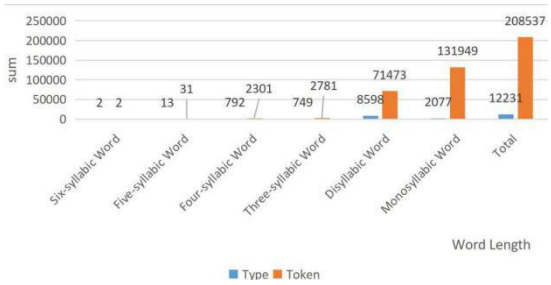


Figure 3: Distribution of word length.

4.2 Parts of speech

Table 1 illustrates the distribution of word length in detail. The CCPL corpus contains a total of 33 POSes. The verbs take vast majority (23.56%), followed by noun (18.09%). On the bottom, prefix amount to a rather low ratio, nearly zero. In addition, there are a considerable number of idioms, phrases, onomatopoeias and so on in the corpus, which reflect the feature of both spoken and written discourse in lyrics.

POS	Token	Ratio	POS	Token	Ratio	POS	Token	Ratio
(verb) v	49133	23.56%	(conjunction)c	2790	1.34%	(phrase)l	255	0.12%
(noun) n	37724	18.09%	(directional verb) vf	2648	1.27%	(other proper noun) nz	165	0.08%
(pronoun) r	23845	11.43%	(exclamation)e	2591	1.24%	(space)s	132	0.06%
(auxiliary) u	20501	9.83%	(modal partical) y	2396	1.15%	(descriptive word) z	129	0.06%
(adjective) a	16870	8.09%	(idiom)i	1789	0.86%	(abbreviation)j	57	0.03%
(adverb) d	16787	8.05%	(noun of place) ns	1489	0.71%	(verb & noun) vn	55	0.03%
(prepositional)p	7829	3.75%	(substantival morpheme) ng	856	0.41%	(adjective & adverb) ad	30	0.01%
(numeral) m	5414	2.60%	(onomatopoeia)o	813	0.39%	(adjective & noun) an	30	0.01%
(time) t	4971	2.38%	(suffix)k	700	0.34%	(verbal morpheme) vg	9	0.00%
(quantity) q	4004	1.92%	(noun of personal name) nr	441	0.21%	(noun of time) nt	7	0.00%
(noun of locality) f	3797	1.82%	(distinguishing words) b	278	0.13%	(prefix)h	2	0.00%

Table 1: Distribution of POS.

4.3 Time and Source Description

Figure 4 illustrates the number of songs collected in the CCPL corpus by the year of publication. It can be inferred that the songs in corpus spans the period from 1990 to 2020.

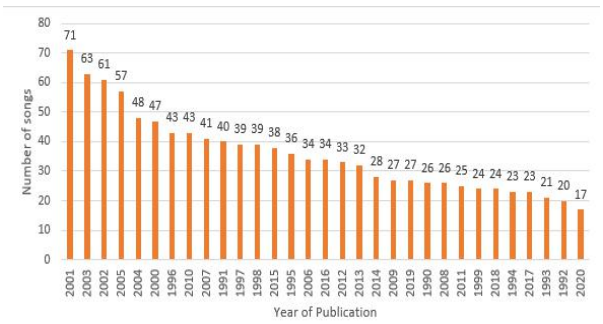


Figure 4: Year of publication distribution.

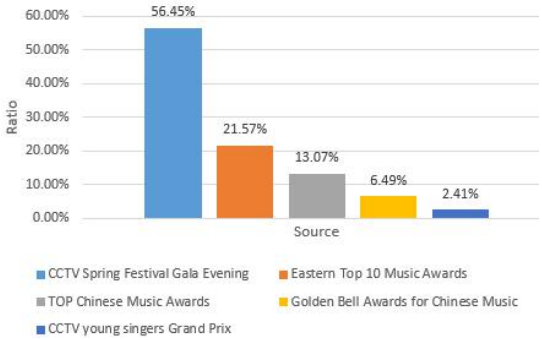


Figure 5: Source distribution.

Figure 5 shows the number of songs published in our corpus by source. We find that over 50% of all songs in the CCPL corpus are from the CCTV Spring Festival Gala Evening and only around 2 are from CCTV young singers Grand Prix.

The CCPL corpus could be downloaded online. We provide the corpus in eXtensible Markup (xml) file formats.[2]

5 Character and Word Analysis

HSK is a standard international Chinese language proficiency test for CFL learners. As an international and authoritative test, it is suitable to be used as a golden standard to measure whether lyrics are appropriate for assisting TCSL. There are 6 levels in the new Syllabus of HSK. CFL learners who reach HSK6 should be competent to master 2,500 Chinese characters and 5,000 words in total.

In this section, repetitive types of characters and words were removed. Character frequency and word frequency were computed to respectively generate the character list and word list for the comparison between the CCPL and HSK in character and word.

5.1 Characters in HSK

Table 2 indicates the character type coverage of the CCPL corpus in HSK. The ratio decreases step by step. The coverage rate at HSK1 is the highest, reaching 100.0% meanwhile that at HSK6 level is the lowest, reaching 84.3%. However, the shared types cover 91.9% of the total 2500 Chinese characters in HSK, indicating that the overall situation of character type coverage is optimistic. Even so, there are 1022 new characters beyond the scope of HSK, among which contains rare characters such as “魁”, “曹”, “桀”, “鸄” that are hard to recognize.

Table 3 describes the character type coverage of HSK in CCPL. It worth to note that types in HSK1 and HSK2 contributes the lowest coverage rate (4.5%) while HSK6 shows the most optimistic situation with the coverage rate of 25.4%. That reveals preliminarily that the presence of new characters is difficult for CFL learners of elementary level (HSK1 and HSK2) in understanding the lyrics content, and when CFL learners reach intermediate level (HSK3 and HSK4) even advanced level (HSK5 and HSK6), the situation will get improved with the superior ability of mastering new characters. It can be inferred that the present of new characters for learners in intermediate level or advanced level may be a good approach to improve the reservation of Chinese characters.

Table 4 describes the character token coverage of HSK in the CCPL corpus. Although the character type coverage of HSK in corpus performs unsatisfying, it can be seen in Table 4 that the 2,298 shared types constitute 96.9% of the content in corpus, in other words, the 1,022 new characters beyond the scope of HSK account for only 3.1% in the corpus. It can also be seen that tokens of HSK 1 and HSK 2 contribute the largest coverage rate, which shows simple characters tend to appear in elementary level. CFL learners at that stage can review the characters they have grasped.

[2] URL: https://pan.baidu.com/s/1txjD9hx2ZlnGYgjDaGxslQ
Password: ykhu

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK 1-6
Shared Types	150	150	298	391	466	843	2298
Types in HSK	150	150	300	400	500	1000	2500
Coverage Rate (Shared Types/HSK)	100.0%	100.0%	99.3%	97.8%	93.2%	84.3%	91.9%

Table2: Shared types in HSK.

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK1-6
Shared Types	150	150	298	391	466	843	2298
Types in CCPL	3320	3320	3320	3320	3320	3320	3320
Coverage Rate (Shared Types/ the CCPL corpus)	4.5%	4.5%	9.0%	11.8%	14.0%	25.4%	69.2%

Table 3: Shared types in CCPL.

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK1-6
Shared Tokens	118080	47009	48958	31910	21024	16489	283470
Tokens in in CCPL	292609	292609	292609	292609	292609	292609	292609
Coverage Rate (Shared Tokens/ the CCPL corpus)	40.4%	16.1%	16.7%	10.9%	7.2%	5.6%	96.9%

Table 4: Shared tokens in CCPL.

5.2 Words in HSK

Table 5 illustrates that the word type coverage in the CCPL corpus in HSK is decreasing step by step, among which the coverage rate in HSK 1 is the highest (93.3%) while that in HSK 6 is the lowest (51.6%). However, the lexical coverage of the CCPL corpus in HSK is not optimistic with the coverage rate of 51.6%. Besides, according to Table 6, the word types of HSK only cover 20.6% in the CCPL corpus, which infers that there is a great amount of words beyond the scope of HSK. Indeed, idioms, proverbs and other kind of phrases can be found in the CCPL corpus, such as “种瓜得瓜, 种豆得豆”, “说闲话”, “鞠躬尽瘁”, etc. Beyond doubt, it is a challenger for CFL learners to master those obscure words without cultural background and language environment.

In table 7, the 2,514 shared word types constitute 67.1% of the content in CCPL in total. It can be inferred that the words beyond the scope of HSK occupies a great part ratio in CCPL, resulting in obstacles in language teaching and learning.

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK 1-6
Shared Types	140	137	248	417	707	938	2580
Types in HSK	150	150	300	600	1300	2500	5000
Coverage Rate (Shared Types/HSK)	93.3%	91.3%	82.7%	69.5%	54.4%	37.5%	51.6%

Table 5: Shared types in HSK.

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK 1-6
Shared Types	136	137	246	406	692	904	2514
Shared Types in CCPL	12231	12231	12231	12231	12231	12231	12231

Coverage Rate (Shared Types/ the CCPL corpus)	1.1%	1.1%	2.0%	3.3%	5.7%	7.4%	20.6%

Table 6: Shared types in CCPL.

	HSK1	HSK2	HSK3	HSK4	HSK5	HSK6	HSK1-6
Shared Tokens	71182	21138	15763	15451	11279	8242	139986
Tokens in CCPL	208537	208537	208537	208537	208537	208537	208537
Coverage Rate (Shared Tokens/ the CCPL corpus)	34.1%	10.1%	7.6%	7.4%	5.4%	4.0%	67.1%

Table 7: Shared tokens in CCPL.

6 Topic Analysis

In this section, for the comparison between lyrics and HSK in topic level, we take the *New General Syllabus for Teaching Chinese* as reference to summarize types of topics discussed in teaching. Meanwhile, we will introduce two weighting schemes with the function of extracting main topics conveyed in lyrics, and provide a more refined classification for the types of lyrics. On the bottom, we conduct a comparative analysis.

6.1 Topics in Textbooks

The teaching topics discussed in practice are diverse and complex, which are difficult to be collected. Zhang Hangli (2019) introduced the topics required in the *New General Syllabus for Teaching Chinese*. Based on that, Zhang classified the topics in *HSK Standard Course*, a textbook with high authority on the market, from the perspectives of topic type and topic content. We took Zhang's study as a standard and divided the passages in 9 copies[3] of *HSK Standard Course* into 16 types of topics, such as *Personal Information*, *Family Life*, *Values*, etc. For instance, *Sunshine always comes after storm* in *HSK Standard Course 4 (I)* and *Reading and Thinking* in *HSK Standard Course 5 (II)* respectively convey the view about success and the attitude towards learning, both of which talk about people's subjective views on specific events. They are classified into *Values* due to the contents we summarize.

6.2 Topics in Lyrics

Removed stop words, we used TF-IDF and Textrank algorithms to uncover the topic addressed in lyrics and divided them into types by the same standard of topics classification in HSK.

6.2.1 TF-IDF algorithm

TF is the abbreviation of Term Frequency, and IDF is the abbreviation of Inverse Document Frequency. The more times a word appears in a particular document and the less it appears in other documents, the more it can reflect the content of the document and the more likely it is to become a key word. Its TF-IDF value is also higher. The formula is shown as follow:

$$TF - IDF = TF * IDF = TFij * log(\frac{N}{Ni})$$

TF_{ij} is the frequency of the term i in document j, N is the total number of documents, and Ni is the number of documents that contain the term i. For instance, top 10 keywords of the song *XiaoFang* uncovered by TF-IDF are "小河", "善良", "谢谢", etc.

6.2.2 Textrank algorithm

The idea of Textrank algorithm is derived from the Pagerank algorithm of Google. Textrank is used to extract keywords, which can be explained by PageRank idea: If a word appears after many words, it means that the word is relatively important. Meanwhile, If a word with a high Textrank value is followed by a word, the

[3] Each level of HSK1-3 is one copy, and each level of HSK4-6 is divided into two copies.

Textrank value of that word is raised accordingly. The formula is shown as follow:

$$S(Vi) = (1-d) + d * \sum(j,i) \frac{Wji}{\sum vk \in out(Vj) Wjk} S(Vj)$$

It can be described as: the weight of a word I in Textrank depends on the weight of the edge (j, i), which is composed of each point j before i, and the sum of the weights of the point j to the other edges. Top 10 keywords of the song *Welcome to Beijing* extracted by Textrank are "欢迎", "北京", "打开", etc.

6.2.3 The Shared Topics in Lyrics and HSK

According to keywords outputted by TF-IDF and Textrank, we removed the repeated units to get a cluster of keywords for each song. Words in clusters have a certain semantic correlation, which could be regarded as the semantic representation of topics. For instance, the word cluster of the song *Ice-sugar gourd* contains 14 words. Several words describe the appearance, such as "竹签", "好看". Some words tell the meaning of this snack, such as "幸福", "团圆". Other words indicate its function, such as "治病". Given that the passage *Radish Cake in hometown* in *HSK Standard Course 5 (II)* discusses about a kind of delicious food and is classified into the type of *Language and Culture*, the topic of *Ice-sugar gourd* is summarized as *Language and Culture* as well. Figure 6 shows the topic distribution in the CCPL corpus.

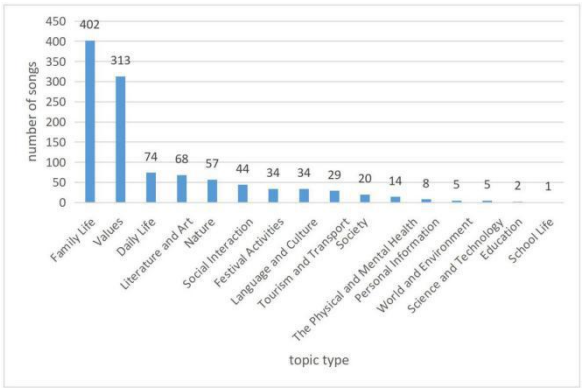


Figure 6: Topic types in the CCPL corpus.

According to Figure 6, *Family Life* takes the vast majority (402), followed by Values (313) and by 10 types of topics (range from 8 to 74). Love songs account for the largest proportion in the CCPL corpus. That is the reason why the topic *Family Life* contributes the largest number. It should be noted that the topic distribution in the corpus is sparse. *Personal Information*, *Science and Technology*, *Education* and *School life* contributes under 5 songs each.

Figure 7, Figure 8, Figure 9 suggests the topic distribution in HSK.

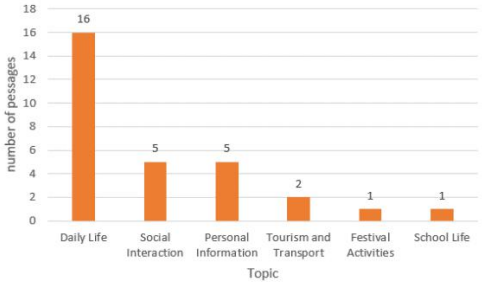


Figure 7: Topic distribution in *HSK Standard Course 1&2.*

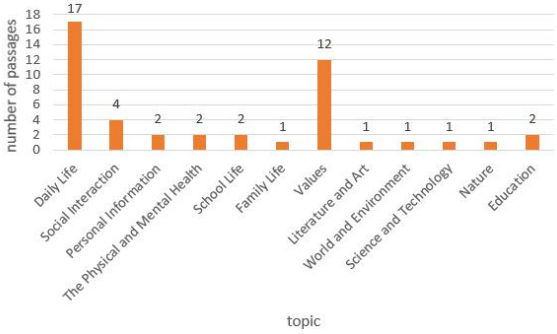


Figure 8: Topic distribution in *HSK Standard Course 3&4.*

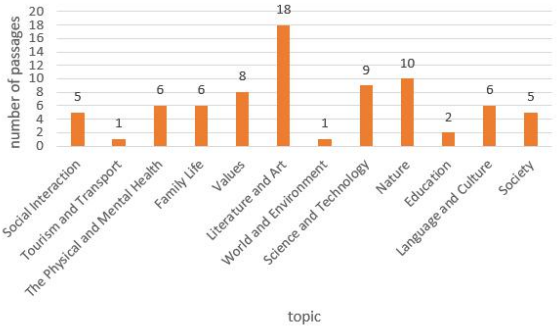


Figure 9: Topic distribution in *HSK Standard Course 5&6.*

As is shown in Figure 7, topics in elementary level tend to be simple and practical, such as *Daily Life*, *Social information* and *Personal information*. They are related to almost all aspects of daily life and include specific interaction environment. And songs classified into these

topics account for a certain number in the CCPL corpus, except *School Life*.

It can be noted in Figure 8 that topics in intermediate level tend to be diverse and difficult, among which begin to involve the outlook and attitude towards health, culture and philosophy, such as *The Physical and Mental Health*, *Values* , *Literature and Art*. Even so, the proportion of *Daily Life* still ranks first. As we can observe with the combination of Figure 6 and Figure 8, teachers have multiple choices when selecting topic-related songs. Meanwhile, they still get limited if the passages in HSK pay attention to *School Life* and *Education*.

According to Figure 9, topics discussed above get involved in advanced level. Different from the levels above, difficult topics account for furtherly more proportion in HSK 5 and HSK6, among which *Literature and Art* ranks first. This also confirms that relevant songs in the CCPL corpus could help broaden students' horizon and cultivate cultural atmosphere.

7 Conclusion and Future Work

In this paper we introduced the CCPL, a homogeneous corpus comprising of 1,110 pop songs. The paper offers the experimental results demonstrating the extent to which pop songs could be appropriate to TCSL teaching. According to the characters coverage, lyrics are suitable for CFL students in elementary level to review the characters they have grasped. As for students in intermediate level and advanced level, they can not only consolidate what they have learned, but also increase their reservation of characters. Lexical coverage is not optimistic. Words beyond the scope of HSK bring barriers in both teaching and learning. Besides, topic coverage confirms the strength of lyrics in introducing cultural background and create multiple language environment, even if teachers may encounter small obstacles in songs selection.

For future work, we intend to give a more refine comparation between lyrics and HSK in character and lexical coverage. The shared and rare characters and words in each song will be calculated. In addition, we will score the appropriateness of each song for language teaching with designing weight formula which includes the factors such as definition, figure of speech, sentence repetition, etc. With relevant information, we can carry out the songs' recommendation for CFL learners at all levels.

8 Acknowledgements

We thank Gaoqi Rao for contributing a lot in consultation and bidding and Mengyao Suo in data selection.

This study was supported by the projects from National Language Committee Project (YB135-90) and BLCU supported project for young researchers program (supported by the Fundamental Research Funds for the Central Universities) (20YCX150).

References

Chenxu Yang. 2019. The Implement of Chinese Music in Teaching Chinese as a Foreign Language. *Today's Massmedia*, 27(03), pages 148-151.

Hangli Zhang. 2019. *A Study on the Distribution of Topic and Content of International Curriculum for Chinese Language Education in HSK Standard Course*, Master's Themes. Sichuan International Studies University, Chongqing, China.

Liping Jiang. 2014. *HSK Standard Course 1*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2014. *HSK Standard Course 2*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2014. *HSK Standard Course 3*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2014. *HSK Standard Course 4(I)(II)*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2015. *HSK Standard Course 5(I)(II)*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2015. *HSK Standard Course 6(I)*. Beijing Language and Culture Press, Beijing, China.

Liping Jiang. 2016. *HSK Standard Course 6 (II)*. Beijing Language and Culture Press, Beijing, China.

Napier, K. and Shamir, L., 2018. Quantitative Sentiment Analysis of Lyrics in Popular Music. *Journal of Popular Music Studies*, 30(4), pages 161-176.

Plitsch, A. 1997. Music + Song = Authentic Listening in the Language Classroom. *In Der Fremdsprachliche Unterricht Englisch*. 31 (1), pages 4–13.

Roman Schneider. 2020. A Corpus Linguistic Perspective on Contemporary German Pop Lyrics with The Multi-Layer Annotated «Songkorpus». In proceedings of LREC 2020, *the 12th Conference on Language Resources and Evaluation*. Marseille, French, 11-16 May, pages 842-848. https://www.aclweb.org/anthology/2020.lrec-1.105/

Shouhui Zhao, Qingsong Luo. 1994. Singing is introduced into Chinese class. *Chinese Language Learning*,1994(4), pages 47-51.

Shiwen Yu, Huiming Duan, Xuefeng Zhu, Bin Sun. 2002.The Basic Processing of Contemporary Chinese Corpus at Peking University. *Journal of Chinese Information Processing*, 2002(5), pages 49-64.

Tegge, F., 2017. The lexical coverage of popular songs in English language teaching. *System*, 2017(67), pages 87-98.

Werner, V., 2012. Love is all around: a corpus-based study of pop lyrics. *Corpora*, 7(1), pages 19-50.

Werner, V. (Ed.), 2018. The language of pop culture. *Routledge Studies in Linguistics 17*. New York: Routledge.

You Fu. 2002. The Introduction of Chinese Songs in Chinese Listening Class. *Journal of Beijing Institute of Education*, 2002(3), pages 64-66.

Yanjing Wang. 2011. The Prevalence of "Sinicism" Songs and Its Application in TCSL. Journal of Sichuan University of Science & Engineering (Social Sciences Edition), 2011(5), pages 86-89.

Author Index

Bade Shrestha, Birat, 114
Bal, Bal Krishna, 114
Bao, Zuyi, 44

Cao, Yongchang, 49
Chao, Rui, 97
Che, Wanxiang, 36
Cheng, Peng, 78
Cheng, Yong, 108
Chi, Xiangyu, 129
Cui, Xin, 57

Dai, Xinyu, 49
Dong, Jinpeng, 57
Duan, Mofan, 108
Duan, Xingyi, 36
Duan, Yitao, 67

Fang, Meiyuan, 67
Fu, Kai, 67
Fu, Ruiji, 36

Gong, Jiefu, 36
Guo, Lei, 57

Han, Yangchao, 97, 102
Han, Yingjie, 97, 102
Haque, Rejwanul, 11
He, Liang, 49
Homma, Hiroki, 1
Hu, Guoping, 36
Hu, Xiao, 36
Huang, Haotian, 102
Huang, Jin, 67

Komachi, Mamoru, 1, 87

Lee, Bruce W, 20
Lee, Jason, 20
Li, Chen, 44
Liang, Deng, 57
Liu, Ting, 36
Liu, Yang, 67
Luo, Yan‘gen, 78
Luo, Yikang, 44

Moslem, Yasmin, 11

Qin, Yufang, 78

Rao, Gaoqi, 25, 129
Ridley, Robert, 49
Rong, Hengqiao, 57

Shen, Zizhuo, 36

Wang, Baoxin, 36
Wang, Hongfei, 87
Wang, Jiping, 67
Wang, Junwei, 91
Wang, Lihuan, 78
Wang, Rui, 44
Wang, Shaolei, 36
Wang, Shijin, 36
Wang, Yi, 78
Wang, Yuke, 102
Wang, Zhongyuan, 36
Way, Andy, 11
Wu, Dayong, 36
Wu, Shih-Hung, 91

Xiong, Xiuzhang, 57

Yan, Yingjie, 97, 102
Yang, Erhong, 25
Yuan, Ruibin, 78
Yue, Gang, 36

Zan, Hongying, 97, 102
Zhang, Baolin, 25
Zhang, KeXin, 121
Zhang, Xuan, 121
Zhang, Yiyang, 121
Zhao, Huizhou, 121
Zheng, Chen, 57
Zhu, NianYong, 78