The LAIX Systems in the BEA-2019 GEC Shared Task

Ruobing Li[†] **Chuan Wang** **Yefei Zha** **Yonghong Yu** **Shiman Guo** **Qiang Wang**
Yang Liu[†] **Hui Lin**[†]

LAIX Inc.

[†]{ruobing.li, yang.liu, hui.lin}@liulishuo.com

Abstract

In this paper, we describe two systems we developed for the three tracks we have participated in the BEA-2019 GEC Shared Task. We investigate competitive classification models with bi-directional recurrent neural networks (Bi-RNN) and neural machine translation (NMT) models. For different tracks, we use ensemble systems to selectively combine the NMT models, the classification models, and some rules, and demonstrate that an ensemble solution can effectively improve GEC performance over single systems. Our GEC systems ranked the first in the Unrestricted Track, and the third in both the Restricted Track and the Low Resource Track.

1 Introduction

Grammatical error correction (GEC) is the task of automatically correcting grammatical errors in text. With the increasing number of language learners, GEC has gained more and more attention from educationists and researchers in the past decade. The following is a GEC example: *I [**fall** → fell] asleep at 11 p.m. last [**nigh** → night].* Here *fall* needs to be corrected to its past tense form and *nigh* is a spelling mistake.

GEC is considered as a mapping task from incorrect sentences to correct sentences. Incorrect sentences can be seen as being produced by adding noises to correct sentences. The added noise does not happen randomly, but occurs when people learn or use the language according to a certain error distribution and language usage bias. Initially, people used rule-based approaches to solve GEC problems (Naber and Miłkowski, 2005). Rules are relatively easy to make but with poor generalization. Later researchers began to treat GEC as a classification task. According to the grammatical information around the target word, classifiers can be constructed to predict the true grammatical role of the target word. One drawback of the classification methods for GEC is that training different classifiers for different error types may be resource-intensive and inefficient since there are many grammatical error types. Recently, translation methods have become the focus of research, and there is a clear trend that state-of-the-art GEC systems are being shifted from traditional NLP methods to NMT based methods.

In recent years, GEC performance has seen significant improvement in some public GEC test sets (Ge et al., 2018). In CoNLL-2013 (Ng et al., 2013) and CoNLL-2014 (Ng et al., 2014) GEC Shared Task, machine learning based GEC methods emerged with relatively good performance. Classification methods achieved the best result in CoNLL-2013 (Rozovskaya et al., 2013). After that, statistical machine translation (SMT) methods began to show better performance in CoNLL-2014 (Felice et al., 2014). (Chollampatt et al., 2016) was the first study to obtain the state-of-the-art result with neural networks. Then after (Junczys-Dowmunt and Grundkiewicz, 2016), machine translation methods became the mainstream in GEC solutions. In addition, an RNN-based context model achieved better results than previous traditional classification models (Wang et al., 2017). Using a CNN-based sequence-to-sequence architecture (Gehring et al., 2017), (Chollampatt and Ng, 2018) proposed the first end-to-end NMT model and reported the state-of-the-art result. As Transformer (Vaswani et al., 2017) plays an increasingly important role in sequence modeling, Transformer-based end-to-end NMT models began to lead the current GEC research (Junczys-Dowmunt et al., 2018; Grundkiewicz and Junczys-Dowmunt, 2018; Ge et al., 2018; Zhao et al., 2019). It is worth mentioning that (Lichtarge et al., 2019) used Wikipedia ed-

Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications, pages 159–167
Florence, Italy, August 2, 2019. ©2019 Association for Computational Linguistics

its history corpus, which is huge but noisy, and gained a result very close to the state-of-the-art result. Learning a GEC translation model from noisy data is a worthy future direction as the GEC parallel corpus is expensive to obtain.

This paper describes our two systems for the three tracks in the BEA-2019 GEC Shared Task (Bryant et al., 2019). We use two popular NMT models and two improved versions of neural classification models to train the basic models. Ensemble strategies are then used to combine outcomes from different models. Our two systems for the three tracks are described in next section. In Section 3, we evaluate the systems on the development data and show the final results on the test data. Section 4 concludes the paper and summarizes the future work.

2 System Overview

2.1 Restricted and Unrestricted Track

We submitted the same system output for the Restricted and Unrestricted tasks. The system uses several ensemble methods to combine the CNN-based and Transformer-based translation models, described in details below.

2.1.1 CNN-based translation ensemble systems

We found that CNN-based systems obtained the best results for some error types, likely due to some characteristics derived from CNN. We trained four CNN-based ensemble systems, using the model architecture in (Chollampatt and Ng, 2018), but without reranking. Four best combinations to build the ensemble systems were selected. Unlike (Chollampatt and Ng, 2018), we did not use fastText (Bojanowski et al., 2017) to initialize word embeddings because we found no improvement on the development set by doing that. We tuned parameters for the system, such as batch size, word embedding dimension, etc.

2.1.2 Transformer-based translation systems

Transformer is currently considered to be one of the most powerful models for sequence modeling. For GEC, some of the best recent results reported on CoNLL-2014 test set are obtained by Transformer-based translation models. We trained eight Transformer-based translation models in a low resource translation paradigm (Junczys-Dowmunt et al., 2018). We tuned parameters for domain and error adaptation. We also compared the results using 2 GPUs and 4 GPUs as the authors reported the difference in their Github repository[1].

2.1.3 Ensemble methods

We expect to combine these models trained above into a more powerful system through effective ensemble methods. Our ensemble work mainly focuses on rule-based solutions. We will introduce two main modules first.

Confidence Table We can obtain the precision and $F_{0.5}$ metric on each error type through sentence alignment and error type classification by Errant (Bryant et al., 2017). Errant provides performance statistics based on 55 error types and is also the tool used to evaluate this GEC shared task, thus we use the result of *operation and error type span-level* (Bryant et al., 2017) for a model or system as the confidence table.

Conflict Solver We often encounter GEC error conflicts when combining multiple models or systems. For example, *We love played soccer.* One system corrects *played* to *playing*, while another system may correct *played* to *to play*. When two different corrections occur in the same place, we need to consider which one to choose.

We solve this problem in a unified pipeline, which can also be seen as an ensemble way:

(1) We sort each group of conflicting corrections proposed by all the systems in a reverse order of location index and confidence.

(2) We apply three sub-strategies:

- When combining outcomes from different systems, we treat the precision in a confidence table as the confidence. Each correction has its confidence obtained by looking up the precision of the corresponding type of the correction in the table. If two conflicting corrections are the same, we merge them and add α to the confidence of the correction; otherwise, the correction with a lower confidence will be discarded.

- After combining outcomes, if the confidence of a correction is lower than β, the correction is discarded.

- γ is used to distinguish when it is more important to focus on the precision or $F_{0.5}$ of

[1] https://github.com/grammatical/neural-naacl2018

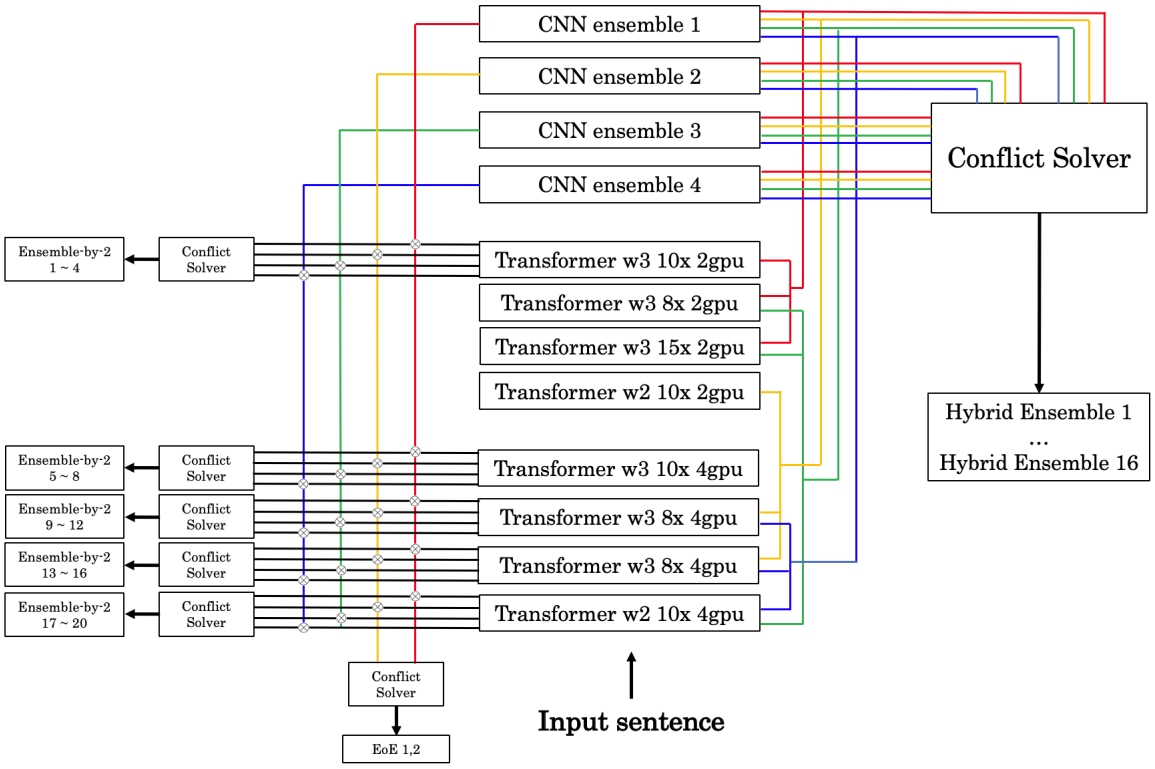

Figure 1: The architecture of the ensemble system in Restricted and Unrestricted Tracks.

a correction. When we move to the final ensemble with confidence tables of existing systems, if the confidence is larger than γ, we select the correction proposed by the system that has the best $F_{0.5}$ on the type of this correction. Otherwise, the correction by a system with the best precision is selected.

In Figure 1, $\otimes$ means the outcome is obtained by combining two systems represented by intersecting lines of two different colours. If there are multiple $\otimes$ on a line, it means the ensemble is over all of these $\otimes$ on this line. Figure 1 displays three types of ensemble methods based on all of the CNN-based and Transformer-based translation models.

- Combine each CNN-based ensemble model with each of the selected five of the Transformer-based models. This is noted as 'ensemble-by-2'.
- Perform ensemble over all of the ensemble models relating to either CNN ensemble 1 or CNN ensemble 2, noted as EoE (Ensemble over Ensemble) 1 and 2.
- Ensemble each CNN ensemble model with some selected combinations of Transformer-based models to produce 16 strong ensemble system outcomes, represented as 'Hybrid Ensemble' in Figure 1. It is where multiple lines of the same color are merged into one line in Figure 1.

After getting all of the ensemble outcomes, we will do the final ensemble step: select the best confidence for each type from each single or ensemble system to form the strongest final outcome. In this ensemble step, we use the last aforementioned sub-strategy, and discard the error types with very low confidence to boost the final performance.

2.2 Low Resource Track

For the Low Resource Track we developed different individual systems and used an ensemble method to combine them. For the translation model, we did not obtain very strong performance because the training data is limited. We also explored the noisy Wikipedia edit history corpus for the Transformer-based translation model. However, we noticed that, for some error types with clear definitions, the classifiers trained on a large amount of native corpus have good performance. In addition, we made some grammatical rules to correct errors and adopted an off-the-shelf spelling checker (Kelly, 2006). Finally, we leverage a sim-

ple ensemble method to combine all of the classifiers, rules, spelling checker and translation models. Note that for the Restricted and Unrestricted tracks, we did not observe any gain from the classification models or the rule-based methods, therefore only the translation systems were used for those tracks.

2.2.1 Classification model

After an analysis of the development sets, we decided to build classifiers for eight common error types. Based on (Wang et al., 2017), we developed two classification model structures for the eight error types.

(A) Bi-GRU context model

Figure 2 shows the bi-directional GRU context model we use to determine the right grammatical category for a target word. The concatenated left and right source states of the target word form the contextual semantic vector representation. This is used as a query to calculate the attention weight a_t. An attention vector C_t is then computed as the weighted average, according to a_t, over all the source states. C_t is then fed through a fully connected layer and softmax layer to produce the predictive distribution.

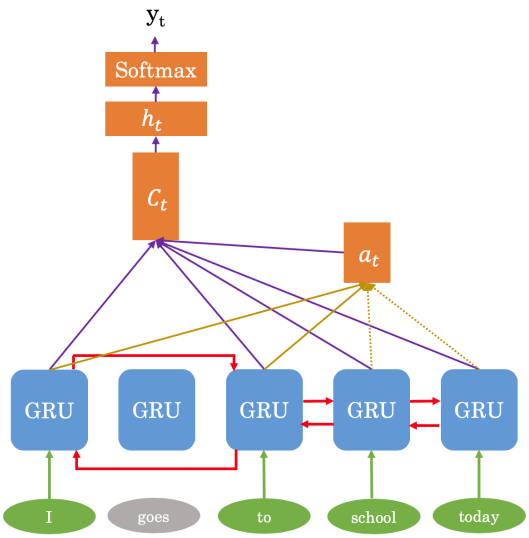

Figure 2: Bi-GRU Context model structure.

We use this to train models for the following error types: Subject-verb agreement, Article, Plural or singular noun, Verb form, Preposition substitution, Missing comma and Period comma substitution. Labels for each task were extracted automatically from the native corpus through part-of-speech tagging tools.

(B) Pointer context model

The classifiers above use the same classification labels for different target words. We also need a classification model to deal with the problem as in the Word form task, where each word has a different set of predictive labels (as shown for word 'gone' in Figure 3). Inspired by the Pointer network model (Vinyals et al., 2015), we proposed the pointer context model. Figure 3 shows the pointer context model that takes the target word's confusion set as the label candidates. The computation path is the same as the Bi-GRU model structure. We concatenate the target word's char-based embedding and C_t to obtain C_t^1, and then use it as the query to compute dot product a_t^1 with each of the word embeddings in the confusion set. a_t^1 is then fed through a softmax layer to produce the predictive distribution. This model is very effective at dealing with varying number of candidates as seen in the Word form task.

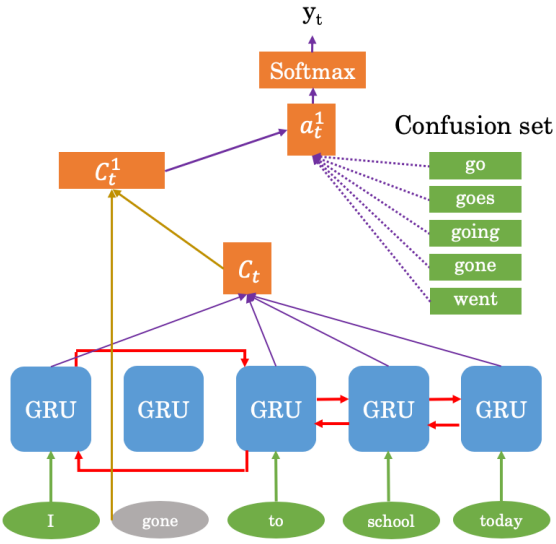

Figure 3: Pointer Context model structure.

2.2.2 NMT model

We use the same Transformer-based translation model mentioned in Subsection 3.2.2. Due to the limitation of the corpus, we leverage the Wiked (Grundkiewicz and Junczys-Dowmunt, 2014) as our training corpus for the NMT model.

2.2.3 Rules and spell checker

We have implemented the following GEC rules.

(1) **'*a*' and '*an*' substitution**. For this problem, we made rules based on the first phoneme of the following word.

(2) **Comma deletion**. After a prepositional

Track	FCE	Lang-8	NUCLE	W&I+ LOCNESS	Common Crawl	Wiked	Wiki dumps
Restricted Track	Yes	Yes	Yes	Yes	Yes	-	-
Unrestricted Track	Yes	Yes	Yes	Yes	Yes	-	-
Low Resource Track	-	-	-	-	Yes	Yes	Yes

Table 1: Corpus used for training in corresponding track.

Seed	Batch size	Word embedding dimension	Number of input channels	Number of output channels	Number of layers	$F_{0.5}$
5001	32	128	256	256	7	0.3370
5002	32	128	256	256	7	0.3219
5003	32	128	256	256	7	0.3370
5004	32	128	256	256	7	0.3411
5005	32	128	256	256	7	0.3449
5012	32	256	512	512	10	0.3339
5102	32	128	512	512	7	0.3329
7011	16	256	512	512	7	0.3328
7205	32	256	512	512	14	0.3328

Table 2: Results of tuned single CNN-based translation models on the development set.

phrase at the beginning of a sentence, we add a comma. For example, *"Despite our differences we collaborate well."* A comma should be added after *Despite our differences*.

(3) **Orthography mistakes**. We obtain statistics of named entities that require initial capitalization and make a white list using the Wikipedia corpus. If a word is on the white list, we will force the conversion to the initial capitalization form.

In addition, we use Pyenchant as our spell checker (Kelly, 2006). The top candidate is considered to be the correction.

2.2.4 Ensemble

We use the conflict solver described above to do the ensemble for all of the outputs of the classifiers, rules, spell checker and NMT model.

3 Experiments

3.1 Data Sets

Table 1 lists the data sets used in **Restricted Track** and **Unrestricted Track**, including FCE (Yannakoudakis et al., 2011), Lang-8[2] (Mizumoto et al., 2012), NUCLE (Ng et al., 2014), W&I+LOCNESS (Bryant et al., 2019) and Common Crawl. We use Common Crawl to pretrain the decoder parameters for the Transformer-based translation model. FCE, Lang-8, NUCLE and W&I are used to train all of the translation models. It is worth noting that we did data augmentation for W&I to train all of the translation models. The data sets used in **Low Resource Track** include Wiked, Wikipedia Dumps and Common Crawl. All of the classifiers are trained on Wikipedia Dumps and the translation model is trained on Wiked corpus. For Wiked corpus, we did some data cleaning work. We discarded some noisy sentences that include error types such as *U:OTHER*, *R:OTHER*, *R:NOUN*, etc. The development set from W&I+LOCNESS are used in all the tracks. Following the data pre-processing pipeline used to generate the data provided by the shared task, we tokenize all of the data using spaCy[3].

3.2 Restricted and Unrestricted Track

3.2.1 CNN-based translation ensemble models

We added the W&I corpus eight times to the training corpus for domain adaptation. Table 2 shows the performance of the single CNN-based translation models. All the parameters in Table 2 are tuned over the W&I+LOCNESS development set.

Table 3 shows the results of the four CNN-based ensemble systems. We use ensembles in the same way as (Chollampatt and Ng, 2018). The above results prove that the ensemble method has yielded a very large improvement in this task.

[2] https://lang-8.com

[3] https://spacy.io

Ensemble index	Combination	Precision	Recall	$F_{0.5}$
1	5012,5102,7011,7205	0.5076	0.2195	0.4021
2	5001,5002,5003,5004	0.5003	0.1951	0.3811
3	5005,5012,5102,7205	0.5156	0.2150	0.4029
4	5005,5012,7011,7205	0.5152	0.2159	0.4034

Table 3: Results of CNN-based ensemble systems on the development set.

Model index	Error weight	Copy number of W&I trainset	GPU number	Precision	Recall	$F_{0.5}$
1	3	10	2	0.4585	0.3525	0.4325
2	3	10	4	0.4602	0.3514	0.4333
3	3	8	2	0.4592	0.3575	0.4345
4	3	8	4	0.4641	0.3548	0.4372
5	3	15	2	0.4494	0.3479	0.4247
6	3	15	4	0.4648	0.3467	0.4352
7	2	10	2	0.4715	0.3303	0.4343
8	2	10	4	0.4868	0.3412	0.4485

Table 4: Results of Transformer-based translation models on the development set.

3.2.2 Transformer-based translation models

We trained eight Transformer-based translation models in different combinations of error adaptation, domain adaptation, and GPU set.

In Table 4, we notice that a smaller error weight yields higher precision and a slight decrease in recall. We set the copy number as 8, 10 and 15, and find that domain adaptation has no significant effect on the results. 4 GPU is obviously better than 2 GPU sets, which is probably because of the larger batch size accumulation for gradient calculation.

3.2.3 Ensemble methods

As described in Section 2.1.3, we need to ensemble all of the CNN-based and Transformer-based translation models. We have already introduced the configuration of the single models in Section 3.2.1 and Section 3.2.2. Next we will describe the configuration of the ensemble system.

For the three ensemble types: Ensemble-by-2, EoE and Hybrid Ensemble, as shown in Figure 1, we used different parameters in the conflict solver.

We did a small-scale grid search for the parameters in Table 5. When combining two models that are not strong, we expect a higher recall so β was not high. For EoE and hybrid ensemble, we expect a higher precision so that they can provide high quality single type performance. Corrections proposed by multiple models are given higher weights (controlled by α). If the confidence of a correction finally reaches β, the correction will be adopted. In the final ensemble, we select the best performance on each type from each single system or ensemble system and discard the corrections with low precision (controlled by β). To get higher $F_{0.5}$, in the case where the precision is greater than a predefined threshold (controlled by γ), we will choose the model with the highest $F_{0.5}$ for the corresponding error type. The final outcome of the data set is then fed through the translation models and ensemble systems again to do a second pass correction.

Ensemble method	α	β	γ
Ensemble-by-2	0.2	0.4	-
EoE	0.15	0.8	-
Hybrid ensemble	0.15	0.62	-
Final ensemble	0.0	0.5	0.52

Table 5: Parameters in the conflict solver for the ensemble methods in **Restricted and Unrestricted Track**.

3.2.4 Results

Table 6 summarizes some results on the development set and gives the official test result. We can see that the individual CNN or Transformer-based translation models perform reasonably well, and the ensemble methods consistently outperform the individual systems. The second pass correction further improves the performance, and the last post-processing step boosts both recall and $F_{0.5}$.

Step	Precision	Recall	$F_{0.5}$
Best CNN-based ensemble model	0.5152	0.2159	0.4034
Best Transformer-based translation model	0.4868	0.3412	0.4485
Best ensemble-by-2	0.5281	0.3434	0.4768
Best hybrid ensemble	0.5885	0.3278	0.5078
+ Combine best performance	0.6283	0.3269	0.5305
+ Second pass	0.6272	0.3412	0.5372
Submission system (+ Post-processing, Dev set)	0.6243	0.3457	0.5376
Submission system (Test set)	0.7317	0.4950	0.6678

Table 6: Results of **Restricted and Unrestricted Track**.

Ensemble method	α	β	γ
Ensemble for all	0.15	0.3	-
Final ensemble	0.0	0.25	0.3

Table 7: Parameters for the ensemble method in **Low Resource Track**.

Table 6 also shows that there is a big gap between the performance on the development set and test set, partly because the final test set uses a combination of five annotators.

3.3 Low Resource Track

3.3.1 Classification models

We trained classifiers for seven error types: Subject-verb agreement, Article, Plural or singular noun, Verb form, Preposition substitution, Missing comma and Period comma substitution and Word form. As mentioned in Subsection 2.2.1, Word form model is trained using the Pointer Context model. The other error types are trained using Bi-GRU Context model.

3.3.2 NMT model

A Transformer-based translation model is trained on the filtered Wiked corpus. The model architecture follows that in (Junczys-Dowmunt et al., 2018). Although the performance of the NMT model is not strong, it provides good performance equivalent to the classifiers for some error types.

3.3.3 Ensemble

We use one conflict solver to combine the outputs from all of the systems in this task. Parameters for this ensemble system are shown in Table 7.

3.3.4 Results

Table 8 shows results for different systems (for classification models, different error type classifiers) on the development set, and the overall results on the test set. We can see that the base systems are not very strong, and the ensemble system significantly improves the performance. The difference between the development set and test set can still be observed in this task.

Model	Precision	Recall	$F_{0.5}$
Rule	0.4497	0.0216	0.0905
Spelling	0.3188	0.0363	0.1248
Article	0.4367	0.0134	0.0597
Missing comma	0.4729	0.0503	0.1763
Period comma substitution	0.4561	0.0070	0.0328
Plural or singular noun	0.3203	0.0121	0.0524
Preposition substitution	0.3713	0.0101	0.0454
Subject-verb agreement	0.3981	0.0115	0.0517
Verb form	0.4135	0.0074	0.0344
Word form	0.4506	0.0294	0.1164
NMT	0.1279	0.1480	0.1315
Submission system (Dev set)	0.4970	0.1686	0.3577
Submission system (Test set)	0.6201	0.3125	0.5181

Table 8: Results of **Low Resource Track**.

4 Conclusions and Future Work

We have presented two different systems for the three GEC tracks. When there is a sufficient parallel learner corpus, such as in **Restricted Track** and **Unrestricted Track**, the NMT ensemble model is the best choice to implement a GEC system. We have evaluated two kinds of NMT models: CNN-based and Transformer-based translation models. We have also explored different ensemble strategies from multiple base mod-

els to maximize the overall system performance. Finally we reached the result of $F_{0.5}$=0.6678 on the official test set in **Restricted Track** and **Unrestricted Track**, ranking the third in the Restricted track[4]. It is worth noting that there is a huge gap between the results on the development set and the test set, which suggests that there might be an unneglectable mismatch between the development set and the test set. Indeed, the development set is annotated by one annotator, while the test set is annotated by five, as announced officially.

For **Low Resource Track**, there is a lack of parallel learner corpus, and thus we rely less on the translation models. We have built eight classifiers trained on Wikipedia dumps according to different error types and an NMT model trained on the Wikipedia edits history corpus. By a simple ensemble method, we reached $F_{0.5}$=0.5181, placing our system in the third place in **Low Resource Track**.

Although GEC has reached the human level performance on some GEC test sets, there is still room for improvement. In a low resource setup, how to deal with the huge but noisy data is worth exploring. (Lichtarge et al., 2019) gave a good solution on this topic, but more work needs to be done. Second, we will investigate methods such as the reinforcement learning based method (Wu et al., 2018) to address the mismatch between the training objectives and evaluation methods in GEC.

References

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The BEA-2019 Shared Task on Grammatical Error Correction. In *Proceedings of the 14th Workshop on Innovative Use of NLP for Building Educational Applications*. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. *In Proceedings of the 32nd AAAI Conference on Artificial Intelligence*.

Shamil Chollampatt, Kaveh Taghipour, and Hwee Tou Ng. 2016. Neural network translation models for grammatical error correction. *In Proceedings of the 25th International Joint Conference on Artifcial Intelligence*.

Mariano Felice, Zheng Yuan, Øistein E. Andersen, Helen Yannakoudakis, and Ekaterina Kochmar. 2014. Grammatical error correction using hybrid systems and type filtering. *Eighteenth Conference on Computational Natural Language Learning*.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching human-level performance in automatic grammatical error correction: An empirical study. *Microsoft Research Technical Report*.

Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N. Dauphin. 2017. Convolutional sequence to sequence learning. *Proceedings of the 34th International Conference on Machine Learning*.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2014. The wiked error corpus: A corpus of corrective wikipedia edits and its application to grammatical error correction. In *Advances in Natural Language Processing – Lecture Notes in Computer Science*, volume 8686, pages 478–490. Springer.

Roman Grundkiewicz and Marcin Junczys-Dowmunt. 2018. Near human-level performance in grammatical error correction with hybrid machine translation. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based machine translation is state-of-the-art for automatic grammatical error correction. *The 2016 Conference on Empirical Methods on Natural Language Processing*.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics*.

Ryan Kelly. 2006. *Pyenchant*.

Jared Lichtarge, Christopher Alberti, Shankar Kumar, Noam Shazeer, and Niki Parmar. 2019. Weakly supervised grammatical error correction using iterative decoding.

Tomoya Mizumoto, Yuta Hayashibe, Mamoru Komachi, Masaaki Nagata, and Yu Matsumoto. 2012. The effect of learner corpus size in grammatical error correction of esl writings. *In Proceedings of COLING 2012*.

Daniel Naber and Marcin Miłkowski. 2005. *LanguageTool*.

[4] https://www.cl.cam.ac.uk/research/nl/bea2019st/#results

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. *Eighteenth Conference on Computational Natural Language Learning*.

Hwee Tou Ng, Siew Mei Wu, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The conll-2013 shared task on grammatical error correction. *Seventeenth Conference on Computational Natural Language Learning*.

Alla Rozovskaya, Kai-Wei Chang, Mark Sammons, and Dan Roth. 2013. The university of illinois system in the conll-2013 shared task. *Seventeenth Conference on Computational Natural Language Learning*.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *31st Conference on Neural Information Processing Systems*.

Oriol Vinyals, Meire Fortunato, and Navdeep Jaitly. 2015. Pointer networks. *Proceedings of the 28th International Conference on Neural Information Processing Systems*.

Chuan Wang, RuoBing Li, and Hui Lin. 2017. Deep context model for grammatical error correction. *Proceedings of the Seventh ISCA workshop on Speech and Language Technology in Education 2017*.

Lijun Wu, Fei Tian, Tao Qin, Jianhuang Lai, and Tie-Yan Liu. 2018. A study of reinforcement learning for neural machine translation. *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading esol texts. *In Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*.

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. *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics*.