On Assessing and Developing Spoken 'Grammatical Error Correction' Systems

Yiting Lu, Mark Gales
ALTA Institute / Engineering Department, University of Cambridge, UK
{ytl28,mjfg100}@cam.ac.uk

Stefano Bannò
Fondazione Bruno Kessler / Department of Cognitive Science, University of Trento, Italy
sbanno@fbk.eu

Abstract

Spoken 'grammatical error correction' (SGEC) is an important process to provide feedback for second language learning. Due to a lack of end-to-end training data, SGEC is often implemented as a cascaded, modular system, consisting of speech recognition, disfluency removal, and grammatical error correction (GEC). This cascaded structure enables efficient use of training data for each module. It is, however, difficult to compare and evaluate the performance of individual modules as preceeding modules may introduce errors. For example the GEC module input depends on the output of non-native speech recognition and disfluency detection, both challenging tasks for learner data. This paper focuses on the assessment and development of SGEC systems. We first discuss metrics for evaluating SGEC, both individual modules and the overall system. The system-level metrics enable tuning for optimal system performance. A known issue in cascaded systems is error propagation between modules. To mitigate this problem semi-supervised approaches and self-distillation are investigated. Lastly, when SGEC system gets deployed it is important to give accurate feedback to users. Thus, we apply filtering to remove edits with low-confidence, aiming to improve overall feedback precision. The performance metrics are examined on a Linguaskill multi-level data set, which includes the original non-native speech, manual transcriptions and reference grammatical error corrections, to enable system analysis and development.

1 Introduction

Grammatical construction is one of the key elements in second language acquisition, and text-based grammatical error correction (GEC) has been widely studied over the past decade (Dale and Kilgarriff, 2011; Ng et al., 2014; Bryant et al., 2017). With speaking skills playing a big part in language learning, it has become increasingly important to analyse spoken grammars. Previous works have investigated grammatical error detection (GED) on spoken language transcriptions (Caines et al., 2020), and tighter integration of disfluency removal and grammar correction on spontaneous learner speech (Lu et al., 2020). This paper focuses on the spoken grammatical error correction (SGEC) task. There are several challenges facing SGEC: running automatic speech recognition (ASR) on learner English is harder than native speech due to potential pronunciation and grammatical errors; spoken language often comes with disfluent speech events such as repetitions and false starts, which are disruptive to downstream tasks; there is very little end-to-end speech to correction data that can be used for training. In this paper, SGEC adopts a cascaded structure: an ASR module produces transcriptions; a disfluency detection (DD) (Zayats et al., 2016) module recovers a fluent text flow; and a conventional machine translation style GEC (Yuan and Briscoe, 2016) module produces error corrections.

Several metrics have been developed to assess text-based GEC. GLEU (Napoles et al., 2015) score adopts BLEU (Papineni et al., 2002) based n-gram precision over the reference. It rewards word-level corrections, as well as correctly preserved source text. MaxMatch M^2 (Dahlmeier and Ng, 2012) captures phrase-level edits, and calculates $F_{0.5}$ scores accordingly. It assesses performance in terms of edits, which suits well with feedback oriented applications. For Spoken GEC assessment, however, it is not straight forward to apply those standard metrics. A common problem facing cascaded style spoken language applications is that it is difficult to compare across models when upstream modules (e.g. ASR) are different. For example, input text to GEC module varies when upstream ASR and DD models are changed. If standard GLEU and M^2 $F_{0.5}$ are to be applied, these metrics mean differently every time ASR transcriptions change, and thus results are incomparable

Proceedings of the 17th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2022), pages 51 - 60

across systems. Not only for cascaded systems, evaluation metrics is not clearly defined for end-to-end trained spoken systems. It is difficult to migrate text-based metrics to spoken tasks, since end-to-end models do not provide any intermediate variables for assessment.

This paper first discusses metrics to assess cascaded SGEC systems. When evaluating individual modules, standard metrics can be used. However, these metrics are not suitable for system-level assessment, since they sometimes take into account module inputs. When downstream module inputs change with its upstream modules, results can become incomparable across systems. To make systems comparable, we use edit distance based metrics instead and focus on the output quality. A common issue in cascaded systems is error propagation, since individual modules are trained separately. To mitigate this issue, semi-supervised fine-tuning is conducted. It aims to tune DD and GEC modules for optimal system performance with non-native ASR transcriptions. Self-distillation is also investigated, which learns from a rich distribution of semi-supervised references. Both fine-tuning experiments are conducted on learner English without readily available annotations. For system development purposes, we focus on optimising output quality; and for assisting language learning, we shift the emphasis to give high quality feedback. We first remove ambiguous corrections, and further filter out low-confidence edits to improve feedback precision as well as the overall quality.

2 Evaluation metrics

Cascaded spoken grammatical error correction (SGEC) consists of three modules, namely speech recognition (ASR), disfluency detection (DD) and grammatical error correction (GEC). It converts disfluent, grammatically incorrect audio sequences into fluent, grammatically correct text. Variables are notated as such: $\boldsymbol{x}$ for input audio, $\boldsymbol{w}$ for speech transcriptions, $\boldsymbol{t}$ for disfluency tags, $\boldsymbol{w}^f$ for transcriptions with disfluencies removed, and $\boldsymbol{y}$ for grammatically correct outputs. N.B.: bold letters are used to represent sequences, with subscripts omitted, e.g. $\boldsymbol{x}$ short for $x_{1:T}$.

$$x_{1:T} \xrightarrow{\text{ASR}} w_{1:N} \xrightarrow{\text{DD}} t_{1:N}, w^f_{1:M} \xrightarrow{\text{GEC}} y_{1:L} \quad (1)$$

When evaluating individual modules, standard metrics can be used. Word error rate (WER) is used for ASR to compute word-level edit distance. DD is modeled as a sequence tagging task, and F_1 score is used to indicate the mean of precision and recall (use hat, e.g. $\hat{a}$, to indicate hypothesises):

$$S_{\text{ASR}} = \text{WER}(\hat{\boldsymbol{w}}, \boldsymbol{w}) \quad (2)$$

$$S_{\text{DD}} = F_1(\hat{\boldsymbol{t}}, \boldsymbol{t}) \quad (3)$$

For GEC module, a standard evaluation is to compare reference and hypothesised edits, and use $F_{0.5}$ score to reflect a weighted precision and recall:

$$E = M^2(\boldsymbol{w}^f, \boldsymbol{y}) \quad (4)$$

$$\hat{E} = M^2(\boldsymbol{w}^f, \hat{\boldsymbol{y}}) \quad (5)$$

$$S_{\text{GEC}} = F_{0.5}(\hat{E}, E) \quad (6)$$

where reference and hypothesised edits E, $\hat{E}$ are extracted using MaxMatch (M^2) (Dahlmeier and Ng, 2012) alignment between inputs and outputs. Each edit is defined by a triplet `[st, ed, cor]` (`st`: start location of the error, `ed`: end location, `cor`: correction).

For cascaded systems, it is also important to look at system-level evaluation, which assesses a combination of modules. When evaluating ASR and DD combined, the standard DD metric F_1 score no longer apply. The reference tags $\boldsymbol{t}$ have a one-to-one correspondence with input word tokens $\boldsymbol{w}$, and we need a different set of reference $\boldsymbol{t}$ for different ASR transcriptions. It is therefore not feasible to compare across systems that have different ASR transcriptions if F_1 is used. Thus, we use WER instead, to directly analyse the output quality from disfluency removal:

$$S_{\text{ASR+DD}} = \text{WER}(\hat{\boldsymbol{w}}^f, \boldsymbol{w}^f) \quad (7)$$

When evaluating ASR, DD and GEC modules combined i.e. the SGEC system, standard GEC metric M^2 $F_{0.5}$ cannot be used, since it does not allow comparison across systems. It requires input sequences w_f to be given for edit extraction, and changes in upstream ASR and DD modules will lead to a different set of reference edits E. Therefore the focus is laid on the quality of outputs. We adopt sentence error rate (SER) to analyse sentence-level matches between references and hypotheses. To achieve greater granularity, we also adopt translation edit rate (TER) (Snover et al., 2006) to assess word-level distance from references.

$$S_{\text{ASR+DD+GEC}} = \text{SER/TER}(\hat{\boldsymbol{y}}, \boldsymbol{y}) \quad (8)$$

Individual module evaluation S_{ASR}, S_{DD} and S_{GEC} helps develop each module separately. System-level metrics $S_{\text{ASR+DD}}$ and $S_{\text{ASR+DD+GEC}}$ both emphasise output quality, which enables comparison across systems even when upstream modules change. They also help guide further tuning and development of the SGEC system as a whole.

3 Module error mitigation

Each module in the cascaded SGEC system is trained individually. DD is trained on a native spoken corpus, and GEC is trained on written text that is processed to be like speech transcripts (details in 5). Individual training allows efficient use of data on one hand, yet on the other hand, it suffers from error propagation due to mismatches between training and evaluation. For example, DD and GEC modules have not seen any ASR transcriptions during training, and thus any ASR error at evaluation time would potentially disrupt their performance. Ideally, fine-tuning on a non-native spoken corpus would most effectively mitigate error propagation, but similar to many other speech to text tasks, there is no readily available data for training. Therefore in this section, we adopt semi-supervised approaches to fine-tune the SGEC system on a spoken learner corpus without manual annotations. Here we use the ASR training corpus, which is comparatively abundant and less costly to obtain, compared to end-to-end SGEC annotation. It consists of audio sequences $\boldsymbol{x}$ and manual transcriptions $\boldsymbol{w}$.

3.1 Semi-supervised fine-tuning

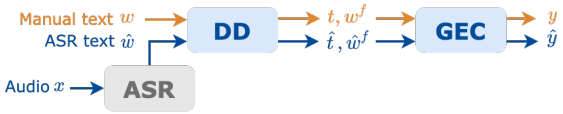

Figure 1: Semi-supervised fine-tuning pipeline. Orange denotes reference generation, blue denotes hypotheses. The greyed (ASR) block is frozen during fine-tuning; DD and GEC modules are separately tuned.

Fine-tuning aims to train the system to be more robust against error propagation. The ASR training set provides non-native audio sequences and their corresponding manual transcriptions, yet lacking references for grammatical error corrections. We generate pseudo references by feeding manual transcriptions through the baseline system, and hypotheses by feeding through audio sequences. The pseudo references are not impacted by ASR errors, and therefore minimising distance between references and hypotheses should help mitigate ASR error propagation. Figure 1 shows the fine-tuning pipeline. When fine-tuning the DD module, references and hypotheses are generated as such:

$$\boldsymbol{t}, \boldsymbol{w}^f = \text{DD}(\boldsymbol{w}) \tag{9}$$

$$\hat{\boldsymbol{t}}, \hat{\boldsymbol{w}}^f = \text{DD}(\text{ASR}(\boldsymbol{x})) \tag{10}$$

Reference tags and fluent text are produced by feeding manual transcriptions through DD module, and hypotheses are generated by feeding audio sequences through ASR and DD modules. For sequence tagging tasks, reference tags change with input word tokens, and thus reference tags $\boldsymbol{t}$ of manual transcriptions cannot be directly applied to ASR transcriptions during fine-tuning. Recalling that WER is used to assess output $\boldsymbol{w}^f$ when evaluating ASR and DD combined, we can apply the same idea and directly compare fluent text $\boldsymbol{w}^f$ and $\hat{\boldsymbol{w}}^f$ after disfluency removal. Reference tags $\boldsymbol{t}'$ can be derived by aligning $\boldsymbol{w}^f$ and $\hat{\boldsymbol{w}}^f$: all insertions in $\hat{\boldsymbol{w}}^f$ are treated as disfluencies, substitutions and matches are tagged as fluent words, whereas deletions are ignored (Table 1):

$\boldsymbol{w}^f$	a	cat	si-	sit	on		the	mat
$\hat{\boldsymbol{w}}^f$	a	cat		sat	on	um	the	mat
Aln	M	M	D	S	M	I	M	M
$\boldsymbol{t}'$	O	O	-	O	O	E	O	O

Table 1: An example of converting $\boldsymbol{w}^f$ and $\hat{\boldsymbol{w}}^f$ to $\boldsymbol{t}'$
M:match, D:deletion, S:substitution, I:insertion
E:disfluent, O:fluent, '-': no label for deleted words

Following this tagging scheme, applying binary cross entropy loss between $\hat{\boldsymbol{t}}$ and $\boldsymbol{t}'$ is equivalent to optimising for lower WER ($S_{\text{ASR+DD}}$). For GEC module fine-tuning, both ASR and DD modules are frozen. To obtain references $\boldsymbol{y}$, manual transcriptions are fed through DD and GEC modules; and for hypotheses $\hat{\boldsymbol{y}}$, audio sequences are fed through ASR, DD and GEC modules:

$$\boldsymbol{y} = \text{GEC}(\text{DD}(\boldsymbol{w})) \tag{11}$$

$$\hat{\boldsymbol{y}} = \text{GEC}(\text{DD}(\text{ASR}(\boldsymbol{x}))) \tag{12}$$

To optimise for a lower SER ($S_{\text{ASR+DD+GEC}}$), a standard cross entropy loss can be used with teacher forcing training:

$$L = \log P(\boldsymbol{y}|\hat{\boldsymbol{w}}^f) = \textstyle\sum_l \log P(y_l|\hat{\boldsymbol{w}}^f, y_{1:l-1}) \tag{13}$$

Minimising cross entropy loss is equivalent to maximising sentence-level probabilities, and therefore should directly help improve SER.

3.2 Self-distillation

Semi-supervised fine-tuning relies on pseudo references generated from manual transcriptions, the quality of which largely depends on the performance of the baseline SGEC system. DD and GEC are trained on native spoken and non-native written corpora respectively, both of which have not encountered any in-domain non-native spoken data. Therefore, it is likely that the pseudo references generated on the non-native spoken corpus are erroneous. To alleviate the potential degradation caused by this, we further apply self-distillation. Self-distillation is originated from knowledge distillation (Hinton et al., 2015), which often trains a student model to learn from predictions made by a teacher model. The teacher is usually superior to the student, e.g. larger in size than the student, or an ensemble teacher for a single model student. Self-distillation (Zhang et al., 2019) is originally proposed in the computer vision community. It extends the idea of knowledge distillation by proposing to use the same model for both teacher and student. It has been shown to be effective for improving both image (Zhang et al., 2019) and text-based tasks (Xu et al., 2020).

Here we adopt the same self-distillation idea, but use a semi-supervised corpus for training. The teacher model is always frozen, and the student model will be trained. The training objective is to minimise the Kullback–Leibler (KL) divergence of the per word posterior distribution between teacher and student:

$$L_{\text{KL}} = \sum_l \text{KL}[P_t(y_l|\hat{\boldsymbol{w}}^f, y_{1:l-1}), P_s(y_l|\hat{\boldsymbol{w}}^f, y_{1:l-1})] \tag{14}$$

where P_t, P_s are teacher and student distributions. Another common practice is to interpolate the KL divergence with cross-entropy loss in Eqn. 13:

$$L_{\text{dist}} = \alpha L_{\text{KL}} + (1-\alpha)L \tag{15}$$

Despite the empirical success of self-distillation, the intuition behind adopting self-distillation on semi-supervised data here is to guide the student model with richer probability distributions, rather than relying solely on one-best predictions.

4 Feedback and confidence filtering

Section 2, 3 mainly focus on evaluating and optimising output quality. Another important aspect for language learning applications is feedback to learners, since feedback quality directly impacts learner's progression in language learning. This section first describes how feedback is extracted and assessed for SGEC systems, then introduces confidence-based filtering that aims to improve feedback precision.

For GEC tasks, feedback usually suggests where the error is and how to correct it. In written GEC, feedback edits are extracted using Eqn. 5 by comparing input and output sequences. Its quality is analysed using $F_{0.5}$ (Eqn. 6) by comparing hypotheses against reference edits. This is not applicable to spoken GEC, since reference edits change with upstream ASR transcriptions, are consequently $F_{0.5}$ scores are not comparable across systems. Here we modify the definition of reference and hypothesised edits as such:

$$E = M^2(\boldsymbol{w}^f, \boldsymbol{y}) \tag{16}$$

$$\hat{E} = M^2(\hat{\boldsymbol{w}}^f, \hat{\boldsymbol{y}}) \tag{17}$$

$F_{0.5}$ can be calculated as before. Reference edits E are generated using manual fluent transcripts $\boldsymbol{w}^f$ as source sequences. With reference E defined independent from ASR or DD module, feedback $F_{0.5}$ can be compared across systems. Hypothesised edits $\hat{E}$ use hypothesised fluent transcriptions $\hat{\boldsymbol{w}}^f$ as source sequences. Therefore $\hat{E}$ account for errors from all three modules, and reflects the true feedback given to users when the system is deployed. Note that such mismatched source sequences in E and $\hat{E}$ put extra penalty on $F_{0.5}$. To given an example: when system output $\hat{\boldsymbol{y}}$ matches with reference $\boldsymbol{y}$ i.e. the system output is perfectly correct, differences in $\boldsymbol{w}^f$ and $\hat{\boldsymbol{w}}^f$ will still result in differences in $\hat{E}$ from E, leading to degraded $F_{0.5}$ score.

SGEC is a very challenging task due to potential errors coming from transcriptions, disfluencies as well as the correction process. To avoid giving erroneous feedback to language learners, we do not want to give feedback on edits that our models have little confidence in, assuming lower confidence indicates lower accuracy. To conduct confidence filtering, we need to define a confidence measure. In the cascaded SGEC pipeline (Eqn. 1), each module produces a token-level confidence score associated with its prediction. We first define sentence-level confidence for each module as the lowest token probability over the entire sentence. Sentence-level filtering can be conducted by rejecting sentences with low confidence. We also explore the option of using edit-level confidence, i.e. confidences are calculated over each edit instead of sentence. Note

that for ASR module, we always use the lowest over sentence, to mitigate a known issue of ASR error propagation. The overall confidence is calculated using a weighted sum of all thee modules:

$$\log P = \alpha \log P_{\text{ASR}} + \beta \log P_{\text{DD}} + \gamma \log P_{\text{GEC}} \tag{18}$$

where P_{ASR}, P_{DD} and P_{GEC} are sentence/edit-level confidence of each module.

5 Experimental results

5.1 Corpora and models

Corpus	Spoken	Use	#Sent	#Word	%Dsf
ASRtrn	✓	ASR Train	62K	2.5M*	-
SWBD	✓	DD Train	154K	940K	11.1
CLC	✗	GEC Train	1.9M	25.2M	0.0
BEA	✗	GEC Train	1M	11.5M	0.0
FCEtst	✗	Eval	2,681	37K	0.0
LIN	✓	Eval	3,361	38K	5.0

Table 2: Corpora statistics. Spoken: whether it is derived from speech; Audio: whether it provides audio sequences; %Dsf: percentage disfluencies contained in the corpus. (*: approximated value, no manual transcriptions available)

ASRtrn is used for ASR training, as well as module error mitigation in Section 3. It consists of 334 hours of an online English speaking test data, which mainly covers 28 L1s and the 5 CEFR (Council of Europe, 2001) grades ranging from A1 to C2. Different from usual ASR training corpus, it only provides crowd source transcriptions, the quality of which is far worse than manual transcriptions. A remedy for this is to use multi-stage teacher-student training: bootstrap the system with crowd source data, and use an ensemble teacher to generate higher quality transcriptions to guide a single student model (Wang et al., 2018). For experiments described in Section 3, by manual transcriptions we always refer to this higher quality transcriptions generated using the teacher ensemble. **Switchboard (SWBD)** (Meteer et al., 1995) consists of 260 hours of telephone conversations of native American English speakers. The Treebank-3 annotation (Taylor et al., 2003) provides manual transcripts and disfluency annotations on the Switchboard corpus. **Cambridge Learner Corpus (CLC)** (Nicholls, 2003) is a collection of written exams of candidates from 86 L1s at different proficiency levels. The corpus is annotated with grammatical errors. **BEA** (Bryant et al., 2019) is a collection of text-based grammatical error correction corpora, including Write & Improve, LOCNESS, Lang-8 and NUCLE (FCE train split excluded, since it overlaps with CLC). **FCEtst** (Yannakoudakis et al., 2011) is a hold out subset of the CLC for test. Punctuation and capitalisation are removed from all corpora derived from written text, to make them look more like speech transcriptions. **Linguaskill (LIN)** is derived from an English speaking test. It consists of 833 learners from over 15 L1s, evenly distributed across CEFR grades. Manual transcriptions are segmented at phrase level, with incomplete or ambiguous phrases rejected. The remaining set is annotated with disfluencies and grammatical errors. Relevant corpora statistics are summarised in Table 2.

Cascaded SGEC consists of three modules: ASR, DD and GEC. **ASR** uses a hybrid deep learning-HMM graphemic system. It is a teacher-student trained TDNN-F model (Povey et al., 2018; Wang et al., 2018) with trigram lattice generation. Succeeding word RNNLM (Chen et al., 2017) is used for rescoring. It has a WER of 19.97% on LIN. Confidence scores are returned by the ASR engines, followed by piece-wise linear mapping (Evermann et al., 2005). **DD** is a binary classification model which consists of a BERT layer (Devlin et al., 2019) in the version provided by the HuggingFace Transformer Library (Wolf et al., 2019) (*bert-base-uncased*), a first dense layer of 768 nodes, a second dense layer of 128 nodes, and finally the output layer of size 2. The model is trained on SWBD and uses an Adam optimiser (Kingma and Ba, 2014) with batch size 64, learning rate 1e-06 and dropout 0.1. **GEC** adopts a transformer-based sequence to sequence model. It is initialised from Gramformer[1], which is a T5 model (Raffel et al., 2020) trained on WikiEdits processed with synthetic error generation techniques (Lichtarge et al., 2019). It is further fine-tuned on CLC and BEA. Training uses Adam optimiser with a batch size of 256, and learning rate of 5e-4 with warm up. Maximum sentence length is set at 64, and the final model parameters are calculated using checkpoint averaging (Izmailov et al., 2018), which takes the average over the 5 best checkpoints. N.B.: all results are reported without standard deviations, since we initialise both DD and GEC modules with large pre-trained models and deviations due to random dropout are relatively small.

[1] https://github.com/PrithivirajDamodaran/Gramformer

5.2 Metrics and tuning

Modules	Metric	In-domain eval Data	In-domain eval Score	LIN-MAN eval Input	LIN-MAN eval Score
ASR	WER ↓	-	-	$\boldsymbol{x}$	19.97
DD	F_1 ↑	SWBD	89.66	$\boldsymbol{w}$	79.52
GEC	M^2 ↑	FCEtst	56.60	$\boldsymbol{w}^f$	53.57

Table 3: Individual module evaluation at their respective operating point. ASR uses LM scale=11, DD uses threshold=0.5.

Table 3 lists performance of each individual module in the SGEC system, comparing out-of-domain evaluation on manual transcriptions of LIN against in-domain test sets. ASR training is conducted in a semi-supervised fashion, therefore we only report WER on LIN-MAN. We always use manual transcriptions for individual module evaluation. For DD, going from native to non-native spoken English, a 10 percent loss is seen in F_1 score. For GEC, going from written to fluent spoken style data loses 3 points on M^2 $F_{0.5}$. Compared to domain mismatch, much larger degradation is induced by ASR errors. Table 4 evaluates combination of multiple modules for LIN, focusing on the overall output quality. It shows significant impact of ASR transcriptions, with the overall SER and TER increasing by 33.50 and 19.62 points respectively.

Modules	Metric	MAN	ASR
ASR+DD	WER ↓	1.96	21.20
ASR+DD+GEC	SER ↓	43.26	76.76
	TER ↓	8.27	27.89

Table 4: Evaluating combination of modules on LIN corpus. MAN and ASR columns show performance on manual and ASR transcriptions at their respective operating points.

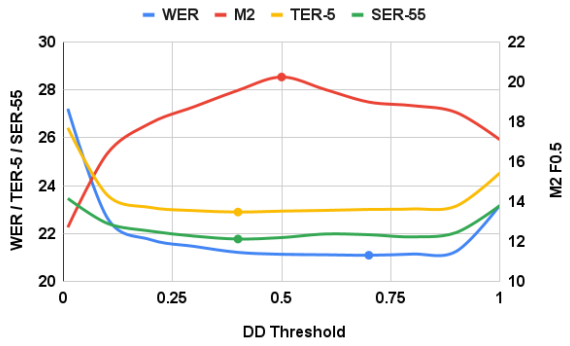

Figure 2: System tuning using SER/TER/M^2. Plot shows a sweep over DD thresholds at the chosen LM scale = 11.

Having defined system level metrics, we can jointly tune the modules in cascaded SGEC for better overall output quality. Here we focus on two variables: ASR language model (LM) scale factor, and DD disfluency removal threshold. A two-dimensional grid search is conducted over a range of LM factors (6-13) and DD threshold (0.0-1.0). Fig. 2 shows a sweep over disfluency removal thresholds at the chosen LM scale factor. It can be seen that all edit distance based metrics are relatively insensitive to the sweep, whereas feedback $F_{0.5}$ shows a stronger preference (more feedback analysis in Section 5.4). The best WER for the intermediate output $\boldsymbol{w}^f$ is at 0.7, and the best SER/TER for the overall output $\boldsymbol{y}$ sits at 0.4. Although differences are insignificant, this shows that an intermediate optima can be different from the overall optima, and confirms the necessity of overall performance metrics. The operating point is chosen according to SER/TER, at a LM scale of 11 and a disfluency threshold of 0.4.

5.3 Module error mitigation

As shown in Table 4, ASR errors result in large degradation, partly because DD and GEC modules have not encounter any non-native spoken data during training. Fine-tuning on a non-native spoken corpus is the most efficient way to mitigate ASR error propagation, yet limited by data availability, we conduct semi-supervised fine-tuning instead. As explained in Section 3.1, we use the SGEC pipeline to generate pseudo reference. Its performance on LIN-MAN (in Table 4) gives an approximation of how much we fall behind supervised fine-tuning. Table 5 lists the impact of tuning DD and GEC modules respectively.

Models	ASR+DD WER↓	ASR+DD+GEC SER↓	ASR+DD+GEC TER↓
Base	21.20	76.76	27.89
TuneDD	21.06	76.79	27.83
TuneGEC	-	76.35	27.47

Table 5: Impact of fine-tuning DD and GEC modules. Note: combination of the two doesn't yield better performance, thus using TuneGEC for future development.

Tuning DD module gives 0.14 decrease on WER of DD output $\boldsymbol{w}^f$, yet fails to improve SER/TER of GEC output $\boldsymbol{y}$. When generating reference tags $\boldsymbol{t}'$ as described in table 1, we minimise the edit distance between $\hat{\boldsymbol{w}}^f$ and $\boldsymbol{w}^f$, which directly optimises for lower WER. However, optimising for the intermediate output does not always help improve the overall output, and thus changes in WER of DD don't seem to have significant impact on downstream GEC. Tuning GEC module improves both SER and TER. The fine-tuning process maximises sentence-level probabilities, which helps to achieve

lower SER/TER.

Semi-supervised fine-tuning of GEC module improves SER/TER, yet it is still not as effective as supervised fine-tuning. Aiming to further improve the output $\boldsymbol{y}$, we adopt semi-supervised self-distillation, which trains the model to learn a probability distribution at each time step, rather than predicting the correct word. The rationale is that probability distribution potentially offers richer information than a single prediction, especially when the reference $\boldsymbol{y}$ is synthetically generated.

Model	Init	KL coeff.	SER↓	TER↓
Teacher	-	-	76.35	27.47
Student	Base	0.5	76.58	27.49
		1.0	76.44	27.46
	Teacher	0.5	76.41	27.46
		1.0	76.35	27.51

Table 6: Self-distillation results. Base and teacher models are Base and TuneGEC from Table 5. Init: initialisation point of the student model. KL coeff.: coefficient of loss interpolation (α in Eqn. 15). Softmax temperature is set at 0.8 for all.

Table 6 contrasts the impact of student initialisation and coefficient of KL loss. The standard approach is to initialise from the teacher, which tends to lead the student to land on a local optima close to the teacher. An alternative is to initialise from Base, which allows the student to explore a larger space, potentially landing on a local minimum further away from the teacher. Larger KL coefficient forces the student to mimic the teacher predicted distribution rather than one-best prediction. However, both SER and TER are quite insensitive to self-distillation, although feedback $F_{0.5}$ shows some improvement (in Section 5.4).

5.4 Feedback and confidence filtering

Previous experiments focus on system analysis and development, this section shift the focus to analyse feedback quality. For optimal feedback, we adopt a slightly different operating point from before according to Fig. 2 (LM scale 11, DD threshold 0.5). Table 7 tabulates the results of system tuning evaluated using system TER and feedback $F_{0.5}$. Compared to system TER, feedback $F_{0.5}$ proves to be much more sensitive to system tuning. Semi-supervised fine-tuning and self-distillation improves feedback by 1.66 and 0.65 points respectively. We use the optimal $F_{0.5}$ (22.57) as our baseline for confidence filtering.

Feedback from SGEC, also called edits, suggests the error location, type and correction. To give high quality feedback to learners, it is important to pass on a clear and accurate message in terms of corrections as well as error types. Feedback edits are automatically typed using a rule-based framework ERRANT (Bryant et al., 2017). Some examples of error types: `M:PREP` (missing preposition), `U:DET` (unnecessary determiner). It sometimes predict error type as `OTHER` when edits do not fall into any other category. A large part of `OTHER` are paraphrases, which can be ambiguous to learners. Therefore we exclude edits typed as `OTHER`.

Models	Base	+Tune	+Distill
TER↓	27.89	27.47	27.46
Fdbk $F_{0.5}$↑	20.26	21.92	22.57

Table 7: Impact of semi-supervised fine-tuning and self-distillation on TER & feedback M^2 $F_{0.5}$. +Distill: self-distillation model initialises from Base, and uses KL coeff.=1.

Table 8 shows that excluding `OTHER` removes approximately 10-15% edits from reference and hypothesis. Note that removing `OTHER` edits in reference reduces the total number of edits, and makes it much easier for models to achieve higher $F_{0.5}$ since most rejected edits are ambiguous and difficult to predict. Both precision and recall get boosted, thus improving the baseline $F_{0.5}$. `OTHER`s are excluded from scoring for confidence filtering experiments below.

	$F_{0.5}$↑		%Edits Exc.	
Eval	Inc.	Exc.	REF	HYP
FCEtst	56.60	59.73	13.87	9.65
LIN	22.57	24.30	14.21	12.54

Table 8: Feedback $F_{0.5}$ inc./exc. `OTHER`, and %edits being removed from reference and hypothesis by excluding `OTHER`.

To improve feedback precision, sentence-level and edit-level confidence filtering are applied to reject ill-conditioned edits. When conducting filtering, we expect both true positives (TP: correctly predicted edits) and false positives (FP: incorrectly predicted edits) to reduce. Under the hypothesis that there are more FPs than TPs in the low confidence region, we expect precision to improve, and consequently help $F_{0.5}$.

Fig. 3 shows change in feedback $F_{0.5}$ score as we filter out an increasing number of edits by setting higher confidence thresholds. Both sentence-level and edit-level filtering peak midway, and drop back as we continue to filter out more edits. Filtering operating at sentence-level tends to work better than edit-level. This can be explained by the na-

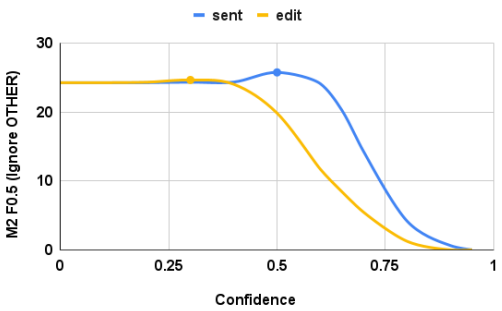

Figure 3: Comparing sentence-level and edit-level confidence filtering. Moving from left to right, confidence threshold increases, and more edits get filtered out.

ture of grammatical corrections being intertwined within one sentence, i.e. removing one edit from a sentence could potentially result in inconsistencies with other corrections made to sentence. Table 9 shows the operating points of confidence filtering. Removing 33.8% of the edits using sentence filtering gives significant gains in both precision and $F_{0.5}$; whereas edit filtering gives mild improvement when 3.7% of edits are filtered out. When deploying SGEC systems, we can always change the confidence threshold to strike a balance between percentage removal and precision improvement.

Filter	P	R	$F_{0.5}$	%Rm
None	27.75	16.24	24.30	0
Sent	33.96	13.15	25.80	33.8
Edit	28.53	16.05	24.69	3.7

Table 9: Operating points of confidence filtering. P: precision, R: recall, %Rm percentage edits being removed

As explained in Eqn. 18, system confidence combines probabilities from all three modules. Fig. 4 analyses the impact of individual modules by contrasting filtering using sentence-level confidence of each module. As an increasing number of edits get filtered out, precision-recall curves move from bottom right to top left corner (precision increases, and recall decreases). Larger area under the curve indicates higher $F_{0.5}$ scores throughout the sweep. Filtering with P_a outperforms both P_d and P_g, suggesting that ASR confidence is quite indicative of feedback quality. Another implication from this observation is that quality of ASR transcriptions largely impacts downstream performances.

We also take a closer look at the impact of sentence-level filtering on different edit types. Table 10 shows the change in precision, recall and $F_{0.5}$ scores before and after filtering. Confidence filtering improves feedback $F_{0.5}$ on most edit types, among which most significant ones are `R:PREP`,

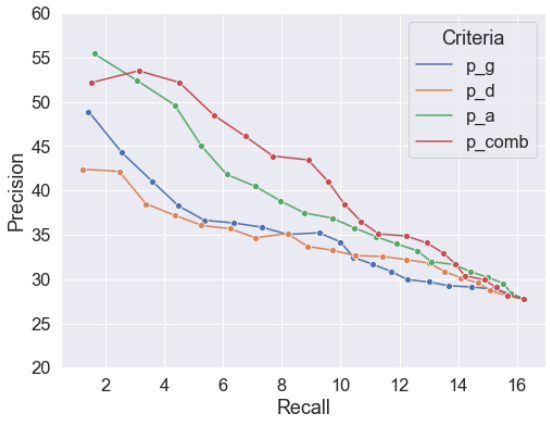

Figure 4: Precision and recall curves: filtering using sentence-level confidences of individual modules. P_{comb}: combined confidence, P_a: ASR, P_d: DD, P_g: GEC. From right to left, filtering out an increasing number of edits. The final α, β, γ coefficients are selected using the optimal $M^2F_{0.5}$ score, which gives $\alpha = 0.3, \beta = 0.4, \gamma = 0.3$.

`U:DET`, `M:PREP`[2]. The two degraded edit types are `R:VERB:TENSE`, `R:VERB:FORM`, both of which often have more than one feasible corrections. Confidence-based filtering tends to remove edits with diverse solutions, due to the high entropy, thus low confidence in the hypotheses. Such rejection pattern leads to significant drop in recall, and consequently reduces $F_{0.5}$ of edits with diverse corrections. On the other hand, for edits like `R:PREP`, `U:DET`, `M:PREP`, there usually exists a single, definite fix. Baseline $F_{0.5}$ scores on those edits are in general quite high, and confidence filtering helps to further improve the performance. Such observation suggests that confidence filtering helps to reduce feedback on ambiguous edits, and in the meantime, boosts precision on more deterministic corrections.

Edit Type	NoFilter P	NoFilter R	NoFilter $F_{0.5}$	Sent P	Sent R	Sent $F_{0.5}$
M:DET	30.18	27.39	29.57	35.86	22.61	**32.10**
R:PREP	37.86	18.47	31.29	46.88	15.68	**33.53**
R:NOUN:NUM	37.88	20.66	32.47	44.68	17.36	**33.98**
R:VERB:TENSE	35.63	13.60	**26.91**	35.00	9.21	22.44
U:DET	23.20	16.20	21.36	29.49	12.85	**23.42**
R:VERB	27.27	11.69	21.53	33.33	10.39	**23.12**
R:NOUN	11.77	4.29	8.72	18.52	3.57	**10.08**
M:PREP	23.29	13.39	20.29	35.56	12.60	**26.06**
R:VERB:FORM	31.17	20.00	**28.04**	38.10	13.33	27.78
R:VERB:SVA	31.92	27.78	30.99	37.31	23.15	**33.25**

Table 10: Comparing $P, R, F_{0.5}$ before and after sentence-level confidence filtering, breakdown by edit types.

[2]There are three prefixes - R: replacement, U: unnecessary, M: missing. The error types are defined using part-of-speech (POS) tags. E.g. `R:PREP` means replacement of preposition. More details on edit types see Bryant et al. 2017.

6 Conclusions

This paper focuses on assessing and developing cascaded SGEC systems. We discuss standard metrics for individual module assessment, as well as edit distance based metrics for system output evaluation. To mitigate module error propagation in cascaded systems, we experimented with semi-supervised fine-tuning and self-distillation approaches, aiming to improve system output quality. Lastly, confidence-based filtering is investigated, and it proves to be effective in improving feedback precision as well as the overall quality.

For future work, we plan to experiment with the state-of-the-art end-to-end ASR systems, which potentially gives lower WER and further improves the SGEC performances. Another research direction is to investigate tighter integration of modular SGEC systems, which allows a richer information flow cross module connections.

Acknowledgements

Thanks to Dr. Linlin Wang for building the ASR system and Dr. Kate Knill for helping generate the ASR confidence scores. This paper reports on research supported by Cambridge Assessment, University of Cambridge. Thanks to Cambridge English Language Assessment for support and access to the Linguaskill data, and Diane Nicholls and her team of ELiT Humannotators for annotating the data set. Work was done while Stefano Bannò was an exchange student at Cambridge.

References

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.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 793–805, Vancouver, Canada. Association for Computational Linguistics.

Andrew Caines, Christian Bentz, Kate Knill, Marek Rei, and Paula Buttery. 2020. Grammatical error detection in transcriptions of spoken english. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2144–2162.

X. Chen, X. Liu, A. Ragni, Y. Wang, and M. J. F. Gales. 2017. Future word contexts in neural network language models. In *Proc. ASRU*, pages 97–103. IEEE.

Council of Europe. 2001. *Common European framework of reference for languages: Learning, teaching, assessment*. Cambridge University Press.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better evaluation for grammatical error correction. In *Proceedings of the 2012 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 568–572, Montréal, Canada. Association for Computational Linguistics.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. 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, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Gunnar Evermann, Ho Yin Chan, Mark JF Gales, Bin Jia, David Mrva, Philip C Woodland, and Kai Yu. 2005. Training LVCSR systems on thousands of hours of data. In *Proceedings.(ICASSP'05). IEEE International Conference on Acoustics, Speech, and Signal Processing, 2005.*, volume 1, pages I–209. IEEE.

Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. 2015. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7).

P Izmailov, AG Wilson, D Podoprikhin, D Vetrov, and T Garipov. 2018. Averaging weights leads to wider optima and better generalization. In *34th Conference on Uncertainty in Artificial Intelligence 2018, UAI 2018*, pages 876–885.

Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. pages 3291–3301.

Y Lu, MJF Gales, and Y Wang. 2020. Spoken language'grammatical error correction'. In *Proceedings of the Annual Conference of the International Speech Communication Association, INTERSPEECH*, volume 2020, pages 3840–3844.

M. Meteer, A. Taylor, R. MacIntyre, and R. Iyer. 1995. Dysfluency Annotation Stylebook for the Switchboard Corpus. Technical report, Linguistic Data Consortium. Updated June 1995 by Ann Taylor.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground truth for grammatical error correction metrics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 588–593, Beijing, China. Association for Computational Linguistics.

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.

Diane Nicholls. 2003. The Cambridge Learner Corpus: Error coding and analysis for lexicography and ELT. In *Proc. of the Corpus Linguistics 2003 conference; UCREL technical paper number 16.*

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, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Daniel Povey, Gaofeng Cheng, Yiming Wang, Ke Li, Hainan Xu, Mahsa Yarmohamadi, and Sanjeev Khudanpur. 2018. Semi-orthogonal low-rank matrix factorization for deep neural networks. In *INTERSPEECH*, pages 3743–3747.

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

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 the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231.

Ann Taylor, Mitchell Marcus, and Beatrice Santorini. 2003. The Penn treebank: an overview. *Treebanks*, pages 5–22.

Y. Wang, J. H. M. Wong, M. J. F. Gales, K. M. Knill, and A. Ragni. 2018. Sequence teacher-student training of acoustic models for automatic free speaking language assessment. *Proc. IEEE Spoken Language Technology Workshop (SLT)*, pages 994–1000.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2019. Huggingface's transformers: State-of-the-art natural language processing.

Yige Xu, Xipeng Qiu, Ligao Zhou, and Xuanjing Huang. 2020. Improving BERT fine-tuning via self-ensemble and self-distillation. *Journal of Computer Science and Technology*, 33:1–18.

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*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386, San Diego, California. Association for Computational Linguistics.

Vicky Zayats, Mari Ostendorf, and Hannaneh Hajishirzi. 2016. Disfluency detection using a bidirectional LSTM. *arXiv preprint arXiv:1604.03209*.

Linfeng Zhang, Jiebo Song, Anni Gao, Jingwei Chen, Chenglong Bao, and Kaisheng Ma. 2019. Be your own teacher: Improve the performance of convolutional neural networks via self distillation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3713–3722.