Taking the Correction Difficulty into Account in Grammatical Error Correction Evaluation

Gotou Takumi[†], Ryo Nagata[†‡], Masato Mita[‡♠], and Kazuhide Hanawa[‡♠]
[†]Konan University
[‡]RIKEN AIP
[♠]Tohoku University
nagata-coling2020-cd @ hyogo-u.ac.jp.

Abstract

This paper presents performance measures for grammatical error correction which take into account the difficulty of error correction. To the best of our knowledge, no conventional measure has such functionality despite the fact that some errors are easy to correct and others are not. The main purpose of this work is to provide a way of determining the difficulty of error correction and to motivate researchers in the domain to attack such difficult errors. The performance measures are based on the simple idea that the more systems successfully correct an error, the easier it is considered to be. This paper presents a set of algorithms to implement this idea. It evaluates the performance measures quantitatively and qualitatively on a wide variety of corpora and systems, revealing that they agree with our intuition of correction difficulty. A scorer and difficulty weight data based on the algorithms have been made available on the web.

1 Introduction

This paper explores difficulty-weighted performance measures for grammatical error correction. The main purpose of this work is to try to increase the diversity of grammatical error correction systems by considering error correction difficulty. In other words, we would like to encourage researchers in the domain in tackling errors that are difficult to correct automatically.

Despite recent progress in grammatical error correction performance, the conventional performance measures such as $F_{0.5}$ and GLEU (Napoles et al., 2015) treat all errors equally. It should be emphasized that some errors are easier to correct than others. For example, errors in spelling[1] are relatively easy to correct. In contrast, errors in *a/the/*ϕ selection and in tense are expected to be much more difficult because it requires wider contextual information or even the intention of the writer to correct them. This nature of the conventional measures discourages researchers from tackling difficult errors. Instead, it encourages them in focusing on frequent errors, which become dominant in the conventional measures. It will be good to have other measures that encourage researchers in attacking difficult errors regardless of frequency. From a technical point of view, it would be interesting to solve difficult problems. Among them, there should be errors that are important in terms of language learning assistance.

Considering this background, this paper takes the first step toward developing performance measures that consider correction difficulty. Generally, one can define the difficulty of a problem in many ways. Our method adopts a simple idea inspired by academic tests (students take, for example). Simply, the more students successfully solve a problem, the easier it is considered to be. This idea can be adopted in grammatical error correction. That is, the more systems successfully correct an error, the easier it is considered to be. In other words, the difficulty of error correction is related to the success rate defined by the number of systems that successfully correct the error in question divided by the total number of systems. Our measures are basically weighted according to this success rate. This will naturally lead to the diversity of grammatical error correction systems because one has to correct errors that others cannot to achieve good performance in these measures.

[1]Spelling errors are included in grammatical errors in the grammatical error correction community.

Proceedings of the 28th International Conference on Computational Linguistics, pages 2085–2095
Barcelona, Spain (Online), December 8-13, 2020

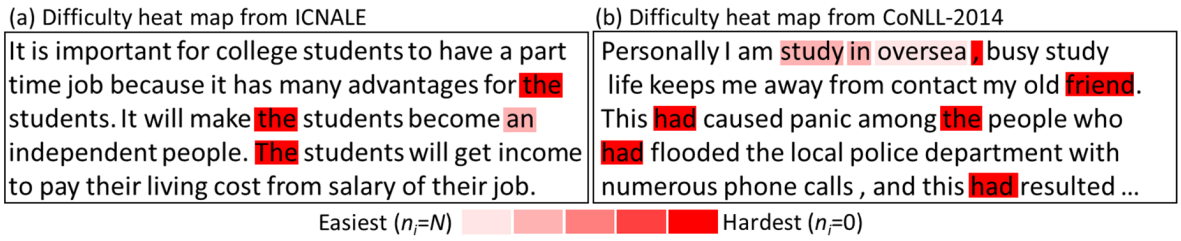


Figure 1: Examples of Difficulty Heat Map.

The contributions of this work are three-fold. First, we present a set of algorithms to calculate the difficulty-weighted measures. It may seem to be trivial to implement the above idea. Contrary to expectation, however, there are some computation problems that need to be solved. Second, we evaluate the measures quantitatively and qualitatively on a wide variety of corpora and systems, revealing their interesting behaviors. Quantitatively, for example, we reveal that they are much more coherent than the conventional $F_{0.5}$ which is known to cause fluctuations in system ranking (Mita et al., 2019). Qualitatively, we show that they agree with the intuitive difficulty of error correction as demonstrated in Fig. 1 where errors are colored according to the success rate: pale (easiest) to deep (hardest) red (details are shown in Sect. 5). In Fig. 1, errors in *a/the/*ϕ selection and tense are recognized as difficult ones while those in spelling and word form as easier ones. Third and finally, we release a tool with difficulty weight data for major evaluation corpora so that anyone can readily evaluate their systems and look into easy and difficult errors[2].

2 Related Work

In grammatical error correction, $F_{0.5}$ (based on recall and precision) and GLEU are normally used as performance measures. In addition, evaluation tools including the MaxMatch (M^2) scorer (Dahlmeier and Ng, 2012) and ERRANT (Bryant et al., 2017; Felice et al., 2016) are available to the public. Without doubt, these measures and tools have greatly contributed to progress in grammatical error correction.

Madnani et al. (2011) propose a performance measure for grammatical error detection. In their measure, recall and precision are weighted according to annotation agreement rates obtained from crowdsourcing in order to take annotation reliability into account. In our measures, recall and precision are also weighted, but according to correction success rates. Therefore, their measure and ours are similar in that both rely on a certain kind of rate. However, ours are different from theirs in that success rates are automatically obtained as system outputs and more importantly, they are used differently as weights; in ours, the lower the success rate is, the higher the weight is, to consider correction difficulty whereas in theirs, it is the other way around to measure annotation reliability.

Numerous corpora are available for grammatical error correction evaluation. These include the CoNLL-2013 (Ng et al., 2013) and CoNLL-2014 (Ng et al., 2014) datasets, Cambridge ESOL First Certificate in English (FCE) (Yannakoudakis et al., 2011), JHU FLuency-Extended GUG Corpus (JFLEG) (Napoles et al., 2017), Konan-JIEM Learner Corpus (KJ) (Nagata et al., 2011), and International Corpus Network of Asian Learners of English, Written Essays (ICNALE) (Ishikawa, 2013). These corpora differ in many aspects: proficiency levels and mother tongues of the writers, essay topics, and error rates to name a few. Evaluation corpora for grammatical error correction have become available for other languages including German (Boyd, 2018), Russian (Rozovskaya and Roth, 2019), Arabic (Mohit et al., 2014), and Chinese (Lee, 2004). Our performance measures can be applied to any corpus in any language as long as it consists of original and correct sentence pairs.

3 Method

3.1 Basic Idea

As already mentioned in Sect. 1, the difficulty of error correction is determined based on the success rate. Take the following sentences as an example:

[2] https://github.com/gotutiyan/GTS

(1) a. Original: He have an aple.
b. Correct: He had an apple.

Suppose that two different systems corrected the original sentence as:

(2) a. Sys1: He had an apple.
b. Sys2: He has an apple.

These corrections would give success rates of 0.5 and 1.0 to the first and second errors, respectively. Then, the difficulty weights for them would be, for example, determined by their reciprocal, resulting in 2.0 and 1.0, respectively.

More generally, the difficulty weight for an error can be defined based on the number of systems successfully correcting it and the total number of systems. To formalize it, let the former and latter numbers be n_i and N, respectively. Then, the weight for the ith error is generally defined by $w_i = f(n_i, N)$. One can think of various functions for f. The above reciprocal is an example. Or, with the hyper parameters a, b, c, $f(n_i, N) = a - \frac{n_i+b}{N+c}$ also satisfies the requirement that the more systems successfully correct the error in question, the easier it is. Hereafter, we limit ourselves to the weight function with $a = 1, b = c = 0$ (i.e., $w_i = 1 - \frac{n_i}{N}$) since it is simple and ranges between 0.0 to 1.0.

This is the basic idea of our performance measures. It is simple and gives incentive to attacking errors that existing systems are not able to correct.

It would be straightforward to count n_i if the lengths of the correct sentence and its corresponding system outputs were always the same as in the above example. In reality, however, this is not the case because correction edits may contain insertions and/or deletions of tokens as in the following example:

(3) Original: We discussing about its .
Correct We have been discussing it .
Sys1: We have been discussing about it .
Sys2: " We are discussing it . "
Sys3: We talking it .

It is then not trivial at all how to count n_i; note that the lengths of all systems have to be the same in order to count n_i properly.

3.2 Calculating Difficulty Weights

This subsection describes the solution to the problem raised in the previous section. For illustration purpose, it often refers to Example (3). In addition, Fig. 2 shows an example flow of the algorithm with a table of the used symbols. Occasionally consulting with them may facilitate understanding this subsection.

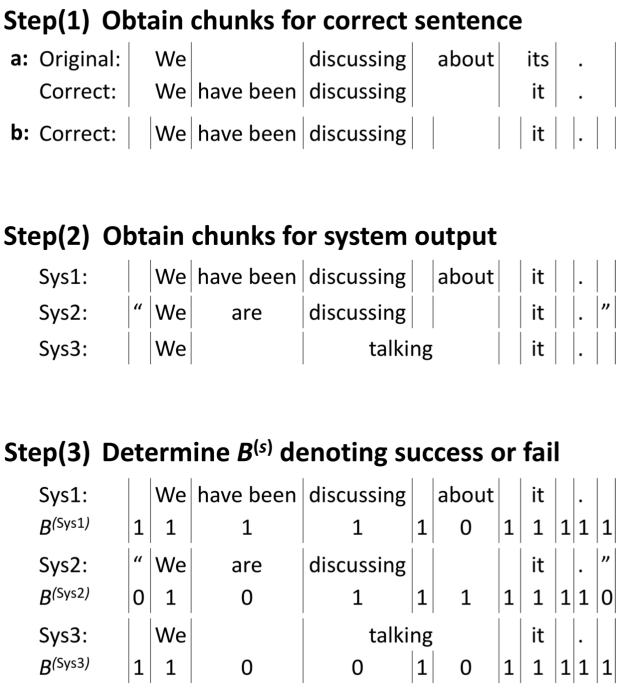
To solve the length problem, the correct sentence is first transformed into a sequence called *chunks* and so are the corresponding system outputs, all of which have the same length. The chunks for each system output are then transformed into a binary sequence denoting whether each correction is successful or not. After this, one can calculate difficulty weights by using the method described in Subsect. 3.1. These procedures are summarized in the following three steps:

Step (1): Transform the correct sentence $T^{(g)}$ into its chunks $C^{(g)}$

Step (2): For all system s, transform its output $T^{(s)}$ into its chunks $C^{(s)}$

Step (3): Transform $C^{(s)}$ into the binary sequence $B^{(s)}$ denoting whether each correction is successful (1) or not (0),

In Step (1), the correct sentence $T^{(g)}$ is transformed into its chunks $C^{(g)}$ by comparing it with the original sentence $T^{(o)}$. First, $T^{(g)}$ is aligned to $T^{(o)}$ by using the alignment algorithm described in the work (Bryant et al., 2017); basically, two sentences are aligned so that it minimizes the Damerau-Levenshtein distance between the two as exemplified in step (1)-a in Fig. 2.



Symbol	Explanation
s	An error correction system
$T^{(o)}$	Tokens in the original sentence
$T^{(g)}$	Tokens in the correct sentence
$T^{(s)}$	Output by the system s
$C^{(g)}$	Chunks for $T^{(g)}$
$C^{(s)}$	Chunks for $T^{(s)}$
$c_i^{(g)}$	ith chunk in $C^{(g)}$
$c_i^{(s)}$	ith chunk in $C^{(s)}$
$B^{(s)}$	Binary sequence denoting whether correction for $c_i^{(s)}$ is successful or not
$b_i^{(s)}$	ith element in $B^{(s)}$
N	# systems
n_i	# systems successfully correcting $c_i^{(g)}$
w_i	Weight for ith chunk
$\mathscr{C}$	Set consisting of indices of $c_i^{(g)}$ aligned to erroneous tokens
$\mathscr{E}$	Set consisting of indices of $c_i^{(g)}$ to which error correction is applied

Figure 2: Flow of Proposed Algorithm with Definition of Symbols.

The boundaries of the aligned tokens (i.e., | in the figure) form the base of $C^{(g)}$. In addition, a dummy chunk is inserted at every boundary of the basic chunks except at those of chunks corresponding to insertion (e.g., the chunk *have been*) so that insertion can be handled; the situation is depicted in Step (1)-b in Fig. 2. The resulting chunks will be denoted by $C^{(g)}$ whose length and ith element will be respectively referred to as M and $c_i^{(g)}$ ($0 \leq i \leq M-1$), hereafter.

In Step (2), each system output $T^{(s)}$ is transformed into its chunks $C^{(s)}$ in the same manner as in Step (1). Each chunk in $C^{(s)}$ will be denoted by $c_i^{(s)}$ as in $C^{(g)}$.

In Step (3), each $C^{(s)}$ is aligned to $C^{(g)}$ so that it minimizes the cost using elastic matching where the cost function assigns 0 if two chunks match or 1, otherwise; roughly, this means that all chunks both in $C^{(s)}$ and $C^{(g)}$ are aligned to any of their counterparts and that there is no crossing between alignments. The match between two chunks $c_i^{(s)}$ and $c_j^{(g)}$ is determined by the following two conditions: (i) all tokens corresponding to the chunks are the same; (ii) the positions of the tokens aligned to the original sentence are the same. For instance, $c_2^{(\text{Sys1})}$ (i.e., *have been*) and $c_2^{(g)}$ (i.e., also *have been*) match because both contain the same tokens and both are aligned to the same position in the original sentence (i.e., between *We* and *discussing*). Step (3) in Figure 2 shows examples of this alignment. The matching results are registered on the basis of $C^{(g)}$. Namely, if $c_i^{(g)}$ matches any chunk in $C^{(s)}$, then 1 is registered; otherwise 0.[3] This procedure gives a binary sequence of the same length as that of $C^{(g)}$ to all $C^{(s)}$. The binary sequence will be denoted by $B^{(s)}$ (with $b_i^{(s)}$ for the ith element).

The fact that all $B^{(s)}$s have the same length M is much more important than it may seem, details of which will be discussed in Subsect. 5.2. Here, let us just mention that from $B^{(s)}$, n_i can be easily counted by $n_i = \sum_s b_i^{(s)}$. For instance, in Example (3), it follows that $n_2 = 1$, $n_5 = 1$, and $n_7 = 3$ and thus, $w_2 = 2/3$, $w_5 = 2/3$, and $w_7 = 0$ (see Step (3) in Fig. 2).

So far, the algorithm has assumed one sentence as input. Without loss of generality, it is applicable to multiple sentences. Simply, Steps (1)–(3) are applied to one sentence at a time.

[3] An exception is that when a chunk in $C^{(s)}$ is aligned to more than one chunks that contain a dummy chunk(s), unmatched dummy chunks are regarded as matched to avoid over-penalizing. This situation is illustrated in the chunk consisting of *talking* in Sys3 and $B^{(\text{Sys3})}$ in Fig. 2.

3.3 Difficulty-Weighted Measures

Now that n_i and w_i are available, they can be applied to conventional measures to obtain their weighted versions. Specifically, the weighted recall and precision are respectively defined by[4]

$$R = \frac{\sum_{i \in \mathscr{E}} w_i b_i}{\sum_{j \in \mathscr{E}} w_j} \tag{1}$$

and

$$P = \frac{\sum_{i \in \mathscr{E}} w_i b_i}{\sum_{j \in \mathscr{E}} w_j b_j + \sum_{k \in \mathscr{C}} w_k (1 - b_k)} \tag{2}$$

where $\mathscr{E}$ and $\mathscr{C}$ denote a set consisting of indices of $c_i^{(g)}$ aligned to an erroneous token(s) in $T^{(o)}$ and another set consisting of indices of $c_i^{(g)}$ to which error correction is applied, respectively[5]. Note that in precision, $1 - b_k$, which corresponds to a false positive, is weighted by w_k. Also note that for the perfect correction (i.e., $b_i = 1$ for all i), $R = P = 1$ is always satisfied no matter how the weight w_i is set as easily demonstrated with $R = \frac{\sum_{i \in \mathscr{E}} w_i \times 1}{\sum_{j \in \mathscr{E}} w_j} = 1$ and $P = \frac{\sum_{i \in \mathscr{E}} w_i \times 1}{\sum_{j \in \mathscr{E}} w_j \times 1 + \sum_{k \in \mathscr{C}} w_k (1-1)} = 1$.

With the weighted recall and precision, one can calculate F_β for an arbitrary choice β. In this paper, $F_{0.5}$ is selected following the convention of research in grammatical error correction.

In addition, this paper introduces a weighted accuracy, which is defined by

$$A = \frac{\sum_{i=0}^{M-1} w_i b_i}{\sum_{j=0}^{M-1} w_j}. \tag{3}$$

This accuracy also satisfies $A = 1$ for any choice of w_i for the perfect correction.

4 Experiments

This section evaluates the proposed performance measures in two ways: cross-corpora evaluation and system-oriented evaluation.

4.1 Cross-corpora Evaluation

The evaluation basically follows Mita et al. (2019)'s work where the following six corpora and four systems are involved: the CoNLL-2013 and CoNLL-2014 test sets, FCE, JFLEG, KJ, and ICNALE; three neural-based methods (encoder-decoder neural networks with a bidirectional LSTM, CNN, or Transformer encoders) and a statistical machine translation-based method (SMT). Table 1 shows the corpus statistics[6]. See their report for other details.

The weighted $F_{0.5}$ and A are calculated for the four systems using the six corpora. Figure 3 shows the results together with the conventional $F_{0.5}$ calculated using the M^2.

Corpus	Number of sentences	Token error rate
CoNLL-2013	1,381	0.15
CoNLL-2014	1,312	0.12
FCE	32,212	0.12
JFLEG	747	0.20
KJ	3,081	0.14
ICNALE	1,732	0.08

Table 1: Statistics on Corpora Used for Experiments.

[4]For readability, the system index s is omitted in the following equations.

[5]One can tell from the alignment results between the original and correct sentences whether a given chunk is erroneous or not. Similarly, one can determine $\mathscr{C}$ by comparing $C^{(s)}$ with $C^{(g)}$ and the alignment results.

[6]For corpora where multiple references were available, we only used the first reference. We will discuss this issue in Subsect. 5.3.

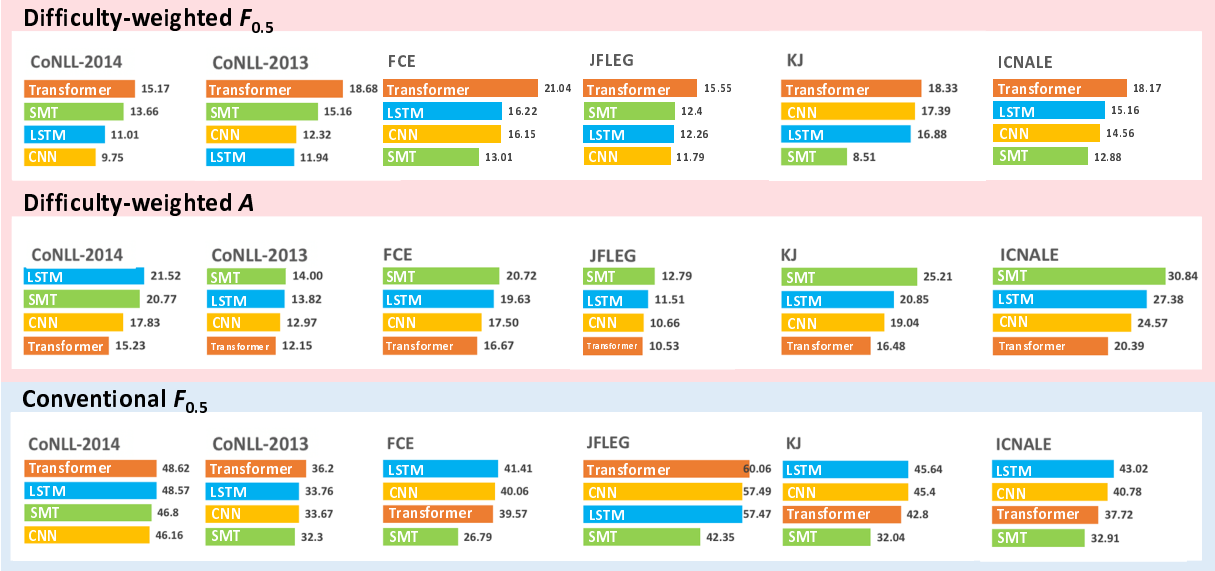


Figure 3: Evaluation in Difficulty-Weighted $F_{0.5}$, Difficulty-Weighted A, and Conventional $F_{0.5}$.

As reported in Mita et al. (2019)'s work, Fig. 3 shows that according to the conventional $F_{0.5}$, the system rankings greatly vary across the six corpora. In contrast, the top-ranked system (Transformer) always remains in first place in our weighted $F_{0.5}$. In addition, the difference in performance between Transformer and the next best systems tend to be large. It would be difficult to tell which measure is better because it depends on the purpose of evaluation. Having said that, the experimental results obtained by the weighted $F_{0.5}$ at least show that Transformer has a different tendency in error correction compared to the others; otherwise it would not have been ranked in first place across all corpora. Subsection 5.2 will discuss this point in more detail.

Figure 3 also shows that the rankings using the difficulty-weighted A are considerably different from those using the weighted $F_{0.5}$ although both are coherent within themselves in that they both rank only one system in first place (except LSTM in A). This seemingly strange phenomenon is explained as follows. It should be first noted that the weighted accuracy involves true negatives while $F_{0.5}$ does not. It should also be noted that correct tokens occupy the majority in all corpora as reflected in the small error rates (see Table 1). Accordingly, true negatives are dominant in accuracy. For this reason, the weighted accuracy favors methods that do not make false positives where other systems do. This is why SMT, whose correction power is limited compared to the others and tends to keep original tokens, ranks in first place most of the time in A. In contrast, $F_{0.5}$ requires an increase in true positives since true negatives have no effect. Thus, unlike A, they favor methods that have the opposite tendency. To sum these up, it is not so bizarre that the rankings obtained by two measures are different. Both measures favor unique systems, but accuracy is more true negative-oriented whereas $F_{0.5}$ is more true positive-oriented. It would be challenging to achieve a good performance in both measures.

4.2 System-oriented Evaluation

To achieve a broader evaluation in terms of systems involved, this evaluation uses correction results of more recent systems including the state-of-the-art one in addition to those of the four used in Subsect. 4.1. The following recent systems, of which outputs for CoNLL-2014 are available on the web, are chosen (accordingly, the target corpus is CoNLL-2014): Kiyono et al. (2019), Junczys-Dowmunt et al. (2018), Junczys-Dowmunt and Grundkiewicz (2016), and Ge et al. (2018). The target measures are the weighted $F_{0.5}$ and A, and the conventional $F_{0.5}$ calculated by the M^2 scorer.

Table 2 shows values of the weighted $F_{0.5}$ together with the difference in rankings compared to the conventional $F_{0.5}$. Table 3 also shows similar data with comparison between the weighted accuracy A and the conventional $F_{0.5}$.

Ranking	System	Weighted $F_{0.5}$	Difference in ranking
1	Kiyono et al. (2019)	26.09	—
2	Junczys-Dowmunt et al. (2018)	21.40	↑ 1
3	Ge et al. (2018)	19.14	↓ 1
4	Junczys-Dowmunt and Grundkiewicz (2016)	15.76	—
5	Transformer	15.05	—
6	SMT	12.78	↑ 1
7	LSTM	10.70	↓ 1
8	CNN	9.81	—

Table 2: Ranking Difference in CoNLL-2014 test set (from Conventional $F_{0.5}$ to Weighted $F_{0.5}$).

Ranking	System	Weighted A	Difference in ranking
1	Kiyono et al. (2019)	25.19	—
2	Junczys-Dowmunt and Grundkiewicz (2016)	24.10	↑ 2
3	Ge et al. (2018)	23.76	↓ 1
4	Junczys-Dowmunt et al. (2018)	22.98	↓ 1
5	LSTM	22.79	↑ 1
6	SMT	21.94	↑ 1
7	CNN	20.13	↑ 1
8	Transformer	17.93	↓ 3

Table 3: Ranking Difference in CoNLL-2014 test set (from Conventional $F_{0.5}$ to Weighted A).

Table 2 and Table 3 show that rankings of some systems change from those given by the conventional $F_{0.5}$ though the differences are not so large. Notably, Junczys-Dowmunt and Grundkiewicz (2016)'s system ranks in second place, gaining two positions. Note that it is based on an SMT, which makes it unique among the other six deep neural-based systems.

It turns out that Kiyono et al. (2019)'s system achieves the best performance in both weighted $F_{0.5}$ and A. As mentioned earlier, it is challenging to do so because of their trade-off relation, and thus it is interesting to discuss why. First of all, they favor a system that is different in terms of errors it corrects. As a matter of fact, Kiyono et al. (2019)'s system is indeed different in that it only exploits pseudo training data generation. Roughly speaking, grammatical error correction systems are normally trained on more or less similar training data such as the CoNLL datasets, which makes systems similar to each other from a training data point of view. Unlike others, Kiyono et al. (2019)'s system uses a large number of correct English sentences and pseudo erroneous sentences (obtained by back translation from the correct sentences). For this reason, it can correct errors that other systems cannot. Besides, the fact that it uses a large number of correct sentences suggests that it knows, through a large number of correct examples, what correct English sentences should be like. As a result, it tends to avoid false positives, resulting in an increase in true negatives (and in turn, in weighted A).

All these findings empirically show that our performance measures evaluate grammatical error correction systems in different ways. More importantly they support the argument that our performance measures favor systems that have different correction tendencies.

5 Discussion

5.1 Revealing Difficult Errors

Now the question is whether our performance measures really reflect error correction difficulty. To answer this question, this subsection first visualizes erroneous chunks by coloring them according to their weights as a heat map: pale (easiest) to deep (hardest) red as already shown in Fig. 1 (on the second page).

Figures 1 (a) and (b) show part of the heat maps obtained from ICNALE with the four systems and

Error Type	Average w_i	SD	Freq.	Error Type	Average w_i	SD	Freq.
ADJ	0.982	0.074	28	WO	0.800	0.265	10
CONTR	0.979	0.051	6	PART	0.797	0.312	16
OTHER	0.972	0.120	440	DET	0.747	0.292	356
PUNCT	0.966	0.114	109	PREP	0.724	0.343	226
CONJ	0.945	0.137	16	MORPH	0.644	0.370	78
ORTH	0.940	0.164	42	VERB:FORM	0.590	0.393	89
NOUN	0.937	0.180	93	NOUN:NUM	0.539	0.340	210
PRON	0.923	0.177	63	SPELL	0.533	0.342	64
ADV	0.911	0.181	55	VERB:SVA	0.499	0.365	97
VERB	0.891	0.254	160	ADJ:FORM	0.375	0.415	5
NOUN:POSS	0.890	0.233	17	NOUN:INFL	0.208	0.246	6
VERB:TENSE	0.876	0.213	191	VERB:INFL	0.062	0.088	2

Table 4: ERRANT's Error Types Sorted by Difficulty Weights (CoNLL-2014, eight systems).

CoNLL-2014 with the eight systems[7]. Figure 1 (a) clearly shows that all systems have difficulty in errors concerning *a*/*the*/ϕ selection (i.e., *the students*). Within a narrow context, the construction would be correct. However, in a broader context, *the students* is incorrect and the definite article should be removed because the writer is talking about students in general, which requires understanding of the discourse of the text and also the intention of the writer. It is highly difficult to correct such errors. Figure 1 (b) shows a similar situation with errors in tense and aspect, which also requires understanding discourse and intention. In contrast, errors requiring only a narrow context are regarded as easier; examples are *an independent people* and *oversea_*, which one can correct without any additional context.

To support the argument, error types, which are automatically obtained by using ERRANT, are sorted by their average difficulty weights obtained from CoNLL-2014 with the eight systems. Table 4 shows the results. As expected, errors involving a narrow context are regarded as easier (e.g., SPELL and VERB:SVA (subject-verb agreement)). In contrast, top rankers are mostly errors concerning lexical choice. Some of them such as ADJ and ADV (adjective and adverb choices, respectively) are relatively infrequent. However, in terms of language learning, it is important to use adjectives and adverbs adequately to write essays with rich descriptions; therefore, in turn, it is important to be able to correct them in language learning assistance. DET and PREP (determiner and preposition errors, respectively) appear in a lower part of the rankings, suggesting that they are rather easier errors. However, their standard deviations are large, which implies that they can be easy and difficult (e.g., *a*/*an* selection vs *a*/*the*/ϕ selection).

5.2 Characteristics of Weighted Measures

The previous subsection has shown that the difficulty weight indeed reflects the difficulty of error correction, at least to some extent. This nature of the difficulty weight (and the weighted measures) brings out the nice property that system rankings according to the weighted measures tend to be stable regardless of evaluation corpora as our experiments have shown. Of course, in theory, they can be variable. However, in practice, the stability is expected to hold unless the distribution of difficult errors changes considerably because difficult errors should appear throughout all proficiencies[8].

Another advantage is that our algorithms solve the problem attributed to the fact that the lengths of the original and correct sentences and also those of system outputs are different. Because of the problem, it has not been trivial how to count the number of instances and thus how to define accuracy of error

[7]Figure 1 excludes omission errors for illustration purposes. Our tool is provided with functions to visualize all three types with information on corrections (also, false positives). The full heat maps are available in the accompanying data. It might be interesting to take a look at which errors are easy and difficult to correct.

[8]The difficulty here is for human writers while that of our measures is for error correction systems. They are not the same, but they are expected to overlap to some extent.

correction[9]. Under our scheme, however, system outputs are mapped to their corresponding correct sentence through its chunks, which makes their lengths identical. This naturally allows us to count the number of instances and accordingly to define accuracy. This property has the further advantage that a wider variety of statistical tests are applicable to evaluation results.

One can argue that our performance measures can result in counter-intuition rankings when one unique system is compared with other systems that are very similar (or even identical) to each other; the unique system might be in first place even if it corrects only a few errors that the others do not.

To discuss this theoretically, let us assume that there are N systems (one of which is a unique system and the rest are identical) and M errors in the target corpus. Further assume that the $N-1$ identical systems correct $100R\%$ of M errors. Then, the weighted recall should be R/NC for the identical systems where C is a certain constant so that recall ranges between 0 and 1; note that the weights for the errors are all $1-(N-1)/N = 1/N$. This means that the unique system would have to correct $MR/(N-1)$ errors that the other systems cannot in order to match them because the weights are all $1-1/N = (N-1)/N$. In other words, the breakpoint is $MR/(N-1)$.

Now, the actual values can be examined with these formulae. For instance, M is about $2,600$ in CoNLL-2014 and R is about 0.45 for the best-performing system (Kiyono et al., 2019). When there are 10 systems (i.e., $N=10$), this follows $MR/(N-1) = 2600 * 0.45/9 = 130$, which amounts to 5% of all 2,600 errors. Therefore, the unique system has to correct 5% of errors that the other systems cannot without affecting the other parts. Whether or not this is better than the performance of the other identical systems can be adjusted by the hyper parameters a, b, c of the weight function. In this paper, we have limited ourselves to $a=1, b=c=0$ (i.e, $w_i = 1 - \frac{n_i}{N}$), which assumes that correcting one error with $w_i = 1$ is equal to correcting two errors $w_i = 0.5$. Under this assumption, the unique system is evaluated to be better than the identical systems when it successfully corrects more than 5% of errors the others cannot. With a higher error of b, for example, the unique system would have to successfully correct more errors to beat the others. It requires more investigation to find the best settings, which is beyond the scope of this paper and will be our future work.

5.3 Limitations of Weighted Measures

One of the limitations is that our performance measures require multiple system outputs. The requirement is naturally satisfied in shared tasks. For this reason, our performance measures are well-suited for the use in shared tasks. Besides, we have released difficulty weight data for the six corpora so that anyone can readily evaluate one's system.

A more crucial problem is that the value of the difficulty weight varies with respect to the number of systems involved in evaluation. This makes it difficult to compare evaluation results involving different systems. Generally speaking, the problem is how to select systems for comparison. It would probably be best to have a standard set of systems for evaluation and to renew the set occasionally as research goes on. For the time-being, our dataset involving the eight systems can be used for this purpose.

Evaluation with multiple references also poses a problem. Conventional measures such as ERRANT compare the system output in question with multiple references and adopt the one that achieves the best performance. This strategy can be applied to our measures, too. Namely, w_i can be calculated for each reference. However, this leads to the situation that the more references are available, the larger w_i tends to be. Besides, the number of errors varies depending on the adopted references. Our performance measures and also the conventional ones assume that the differences are negligible although strictly, any performance measure calculated from data with different numbers of errors cannot be compared directly. More investigation is needed to solve this problem.

One thing missing in the present work is the correlation between the correction difficulties that the proposed method produces and that human experts estimate; it is preferable that two exhibit a high correlation. Having said that, it is a difficult task even for human experts to accurately estimate correction difficulty. Intuitively, errors that require wider contexts for correction tend to be considered more difficult. Also, error correction tends to be more difficult when other errors appear around the error in

[9]There are at least three possibilities: the numbers of tokens in original/correct sentences, or system outputs.

question. However, it is not straightforward at all to tell which case is more difficult. More generally, it is a problem of how to define correction difficulty. The proposed method solves this problem by simply defining it based on the success rate. The above qualitative analysis suggests that our measures have some correlation with human judgements in terms of correction difficulty. It requires further investigations to confirm this point quantitatively.

As already discussed, our performance measures have several good properties as well as some drawbacks. It would be impossible to achieve the perfect evaluation with only one performance measure; it depends on the purpose of evaluation. Accordingly, it is better to have various performance measures so that we can select suitable ones depending on their purposes. The evaluation results and the discussion have shown the unique properties of our performance measures.

6 Conclusions

This paper has taken the first step toward developing performance measures that consider correction difficulty to encourage researchers to tackle more difficult errors. It first introduced the basic idea that the more systems successfully correct an error, the easier it probably is. It then described a set of algorithms to implement the idea as difficulty-weighted performance measures. It showed empirically that they reflect the difficulty of error correction, at least to some extent, which gives incentive to tackling more difficult errors. It further discussed their characteristics and limitations.

In future work, we will evaluate the correlation between the correction difficulties that the proposed method produces and that human experts estimate. Also, we will investigate how we can solve the problem in the use of multiple references.

Acknowledgements

We would like to thank the three anonymous reviewers for their useful comments on this paper. We also would like to thank Shun Kiyono and Tomoya Mizumoto who provided us with some of the error correction results used in the evaluation.

References

Adriane Boyd. 2018. Using Wikipedia edits in low resource grammatical error correction. In *Proceedings of the 2018 EMNLP Workshop W-NUT: The 4th Workshop on Noisy User-generated Text*, pages 79–84.

Christopher Bryant, Mariano Felice, and Ted Briscoe. 2017. Automatic annotation and evaluation of error types for grammatical error correction. In *Proceedings of ACL (Volume 1: Long Papers)*, pages 793–805, July.

Daniel Dahlmeier and Hwee Tou Ng. 2012. Better Evaluation for Grammatical Error Correction. In *Proceedings of NAACL*, pages 568–572.

Mariano Felice, Christopher Bryant, and Ted Briscoe. 2016. Automatic extraction of learner errors in ESL sentences using linguistically enhanced alignments. In *Proc. of 26th International Conference on Computational Linguistics: Technical Papers*, pages 825–835, Osaka, Japan.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching Human-level Performance in Automatic Grammatical Error Correction: An Empirical Study. *arXiv*.

Shin'ichro Ishikawa. 2013. The ICNALE and Sophisticated Contrastive Interlanguage Analysis of Asian learners of English. *Learner Corpus Studies in Asia and the World*, 1:91–118.

Marcin Junczys-Dowmunt and Roman Grundkiewicz. 2016. Phrase-based Machine Translation is State-of-the-Art for Automatic Grammatical Error Correction. In *Proceedings of EMNLP*, pages 1546–1556.

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 NAACL*, pages 595–606.

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 EMNLP-IJCNLP*, pages 1236–1242.

John Lee. 2004. Automatic article restoration. In *Proc. of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics*, pages 31–36.

Nitin Madnani, Martin Chodorow, Joel Tetreault, and Alla Rozovskaya. 2011. They can help: Using crowdsourcing to improve the evaluation of grammatical error detection systems. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 508–513.

Masato Mita, Tomoya Mizumoto, Masahiro Kaneko, Ryo Nagata, and Kentaro Inui. 2019. Cross-corpora evaluation and analysis of grammatical error correction models — is single-corpus evaluation enough? In *Proceedings of NAACL: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 1309–1314, June.

Behrang Mohit, Alla Rozovskaya, Nizar Habash, Wajdi Zaghouani, and Ossama Obeid. 2014. The first QALB shared task on automatic text correction for Arabic. In *Proceedings of the EMNLP 2014 Workshop on Arabic Natural Language Processing (ANLP)*, pages 39–47.

Ryo Nagata, Edward Whittaker, and Vera Sheinman. 2011. Creating a manually error-tagged and shallow-parsed corpus. In *Proceedings of ACL*, pages 1210–1219.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2015. Ground Truth for Grammatical Error Correction Metrics. In *Proceedings of ACL*, pages 588–593.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A Fluency Corpus and Benchmark for Grammatical Error Correction. In *Proceedings of EACL*, pages 229–234.

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 CoNLL 2013 Shared Task*, pages 1–12.

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 CoNLL 2014 Shared Task*, pages 1–14.

Alla Rozovskaya and Dan Roth. 2019. Grammar error correction in morphologically rich languages: The case of Russian. *Transactions of the Association for Computational Linguistics*, 7:1–17, March.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A New Dataset and Method for Automatically Grading ESOL Texts. In *Proceedings of ACL*, pages 180–189.