Towards standardizing Korean Grammatical Error Correction: Datasets and Annotation

Soyoung Yoon
KAIST AI
soyoungyoon@kaist.ac.kr

Sungjoon Park
SoftlyAI Research, SoftlyAI
sungjoon.park@softly.ai

Gyuwan Kim
UCSB[*]
gyuwankim@ucsb.edu

Junhee Cho
Google
junheecho@google.com

Kihyo Park
Cornell University
kp467@cornell.edu

Gyutae Kim
SoftlyAI
gt.kim@softly.ai

Minjoon Seo
KAIST AI
minjoon@kaist.ac.kr

Alice Oh
KAIST SC[†]
alice.oh@kaist.edu

Abstract

Research on Korean grammatical error correction (GEC) is limited, compared to other major languages such as English. We attribute this problematic circumstance to the lack of a carefully designed evaluation benchmark for Korean GEC. In this work, we collect three datasets from different sources (Kor-Lang8, Kor-Native, and Kor-Learner) that covers a wide range of Korean grammatical errors. Considering the nature of Korean grammar, We then define 14 error types for Korean and provide KAGAS (Korean Automatic Grammatical error Annotation System), which can automatically annotate error types from parallel corpora. We use KAGAS on our datasets to make an evaluation benchmark for Korean, and present baseline models trained from our datasets. We show that the model trained with our datasets significantly outperforms the currently used statistical Korean GEC system (Hanspell) on a wider range of error types, demonstrating the diversity and usefulness of the datasets. The implementations and datasets are open-sourced.[1]

1 Introduction

Writing grammatically correct Korean sentences is difficult for learners studying Korean as a Foreign Language (KFL) and even for native Korean speakers due to its morphological and orthographical complexity such as particles, spelling, and collocation. Its word spacing rule is complex since there are many domain-dependent exceptions, of which only around 20% of native speakers understand thoroughly (Lee, 2014). Since Korean is an agglutinative language (Sohn, 2001; Song, 2006), getting used to Korean grammar is time-consuming

for KFL learners whose mother tongue is non-agglutinative (Haupt et al., 2017; Kim, 2020). However, despite the growing number of KFL learners (Lee, 2018), little research has been conducted on Korean Grammatical Error Correction (GEC) because of the previously described difficulties of the Korean language. Another major obstacle to developing a Korean GEC system is the lack of resources to train a machine learning model.

In this paper, we propose three datasets that cover various grammatical errors from different types of annotators and learners. The first dataset named Kor-Native is crowd-sourced from native Korean speakers. Second, Kor-Learner are from KFL learners that consists of essays with detailed corrections and annotations by Korean tutors. Third, Kor-Lang8 are similar with Kor-Learner except that they consist of sentences made by KFL learners but corrected by native Koreans on social platforms who are not necessarily linguistic experts. We also analyze our datasets in terms of error type distributions.

While our proposed parallel corpora can be served as a valuable resource to train a machine learning model, another concern is about the annotation of the datasets. Most existing datasets do not have annotation, which makes it hard to use them for evaluation. A major weakness of human annotation is that (1) experts specialized in Korean Grammar are expensive to hire, (2) making them annotate a large number of parallel corpora is not scalable, and (3) the error types and schema are different by datasets and annotators, which is counterproductive. Another way that we can analyze and evaluate on the dataset is by automatic annotation from parallel corpora. While there is already one for English called ERRANT (Bryant et al., 2017), there is no automatic error type detection system for Korean. We cannot fully demonstrate and classify error types and edits by using ERRANT, because Korean has many different characteristics

---

[*] University of California, Santa Barbara

[†] School of Computing, KAIST

[1] Code for model checkpoints and KAGAS: https://github.com/soyoung97/Standard_Korean_GEC. Dataset request form: https://forms.gle/kF9pvJbLGvnh8ZnQ6

than English (Section 4.5). This motivates us to develop an automated error correction system for Korean (KAGAS), along with annotated error types of refined corpora using KAGAS.

Lastly, we build a simple yet effective baseline model based on BART (Lewis et al., 2019) trained from our datasets. We further analyze the generated outputs of BART on how the accuracy of each system differs by error types when compared with a statistical method called Hanspell,[2] providing use cases and insights gained from analysis. To summarize, the contributions of this paper are as follows: (1) collection of three different types of parallel corpora for Korean GEC, (2) a novel grammatical error annotation toolkit for Korean called KAGAS, and (3) a simple yet effective open-sourced baseline Korean GEC models trained on our datasets with detailed analysis by KAGAS.

2 Related Work

Datasets Well-curated datasets in each language are crucial to build a GEC system that can capture language-specific characteristics (Bender, 2011). In addition to several shared tasks on English GEC (Ng et al., 2014; Bryant et al., 2019; Rao et al., 2018), resources for GEC in other languages are also available (Wu et al., 2018; Li et al., 2018; Rozovskaya and Roth, 2019; Koyama et al., 2020; Boyd, 2018). Existing works on Korean GEC (Min et al., 2020; Lee et al., 2021; Park et al., 2020) are challenging to be replicated because they use internal datasets or existing datasets without providing pre-processing details and scripts. Therefore, it is urgent to provide publicly available datasets in a unified and easily accessible form with pre-processing pipelines that are fully reproducible for the GEC research on Korean.

Evaluation M^2 scorer (Dahlmeier and Ng, 2012) which measures precision, recall, and $F_{0.5}$ scores based on edits, is the standard evaluation metric for English GEC models. It requires an M^2 file with annotations of edit paths from an erroneous sentence to a corrected sentence. However, it is expensive to collect the annotations by human workers as they are often required to have expert linguistic knowledge. When these annotations are not available, GLEU (Napoles et al., 2015), a simple variant of BLEU (Papineni et al., 2002), is used instead by the simple n-gram matching. Another way of generating an M^2 file for English in a

	Kor-Learner	Kor-Native	Kor-Lang8
# Sentence pairs	28,426	17,559	109,559
Avg. token length	14.86	15.22	13.07
# Edits	59,419	29,975	262,833
# Edits / sentence	2.09	1.71	2.40
Avg. tokens per edit	0.97	1.40	0.92
Prop. tokens changed	28.01%	29.37%	39.42%

Table 1: Data statistics for Kor-Learner, Kor-Lang8, and Kor-Native.

rule-based manner is by using the error annotation toolkit called ERRANT (Bryant et al., 2017). We extend ERRANT to make KAGAS and utilize it to align and annotate edits on our datasets and make an M^2 file to evaluate on Korean GEC models.

Models Early works on Korean GEC focus on detecting particle errors with statistical methods (Lee et al., 2012; Israel et al., 2013; Dickinson et al., 2011). A copy-augmented transformer (Zhao et al., 2019) by pre-training to denoise and fine-tuning with paired data demonstrates remarkable performance and is widely used in GEC. Recent studies (Min et al., 2020; Lee et al., 2021; Park et al., 2020) apply this method for Korean GEC. On the other hand, Katsumata and Komachi (2020) show that BART (Lewis et al., 2020), known to be effective on conditioned generation tasks, can be used to build a strong baseline for GEC systems. Following this work, we load the pre-trained weights from KoBART,[3] a Korean version of BART, and finetune it using our GEC datasets.

3 Data Collection

We build three corpora for Korean GEC: Kor-Learner (§3.1), Kor-Native (§3.2), and Kor-Lang8 (§3.3). The statistics of each dataset is described on Table 1. We describe the main characteristic and source of the dataset and how it is preprocessed in the following subsection. We expect that different characteristics of these diverse datasets in terms of quantity, quality, and error type distributions (Figure 1) allow us to train and evaluate a robust GEC model.

3.1 Korean Learner Corpus (Kor-Learner)

Korean Learner Corpus is made from the NIKL learner corpus[4]. The NIKL learner corpus contains essays written by Korean learners and their gram-
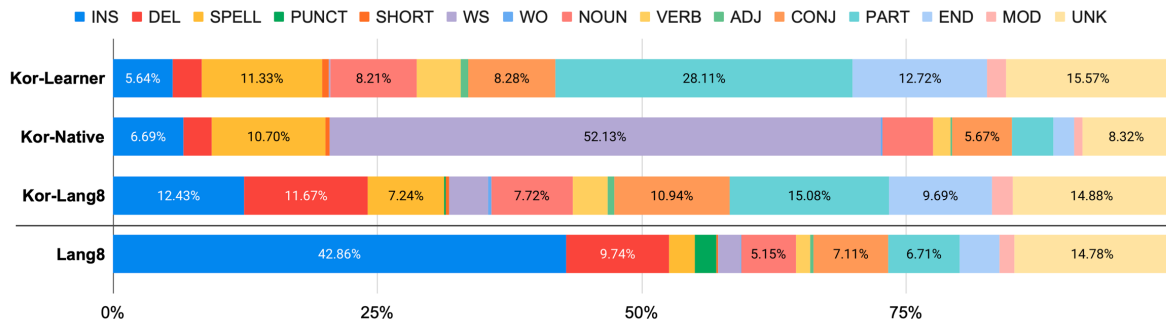

Figure 1: The distribution of error types by our proposed dataset (Upper three). Percentages are shown for error types that has larger than 5%. The bottom row (Lang8) is for comparison with Kor-Lang8. We can see that Kor-Lang8 has fewer outliers and with decrease in INS or DEL edits than the original Lang8 (bottom). The distribution of error types for KFL learners show that NOUN, CONJ, END and PART are frequent than native learners, and word space errors (purple) are the most frequent for native learners (Kor-Native), similar with previous corpus studies (Kim, 2020; Lee, 2020).

matical error correction annotations by their tutors in an morpheme-level XML file format. The original format is described at Appendix A.4.1. Even though the NIKL learner corpus contains annotations by professional Korean tutors, it is not possible to directly be used as a corpus for training and evaluation for two reasons. First, we cannot recover the corrected sentence from the original file nor convert the dataset into an M^2 file format (Section 2) since the dataset is given by *morpheme-level* (syllable-level) correction annotations, not word-level edits. A simple concatenation of morpheme-level edits does not make a complete word since Korean is an agglutinative language. Therefore, we refer to the current Korean orthography guidelines[5] to merge morpheme-level syllables into Korean words (Appendix A.4.3 [6]). Second, some XML files had empty edits, missing tags, and inconsistent edit correction tags depending on annotators, so additional refinement and proofreading was required. Therefore, the authors manually inspected the output of parallel corpora and discard sentences with insufficient annotations (Appendix A.4.2). After applying appropriate modifications to the NIKL corpus, we were able to make Kor-Learner which contains word-level parallel sentences with high quality.

3.2 Native Korean Corpus (Kor-Native)

The purpose of this corpus is to build a parallel corpus representing grammatical errors native Korean speakers make. Because the Korean orthography

guidelines are complicated consisting of 57 rules with numerous exceptions,[5] only a few native Korean speakers fully internalize all from the guidelines and apply them correctly. Thus, the standard approach depends on the manpower of Korean language experts, which is not scalable and is very costly. Thus, we introduce our novel method to create a large parallel GEC corpus from correct sentences, which does not depend on the manpower of experts, but the general public of native Korean speakers. Our method is characterized as a *backward* approach. We collect grammatically correct sentences from two sources,[7] and read the correct sentences using Google Text-to-Speech (TTS) system. We asked the general public to dictate grammatically correct sentences and transcribe them. The transcribed sentences may be incorrect, containing grammatical errors that the audience often makes. Figure 1 shows that most of the collected error types were on word spacing. While the distributions of transcribed and written language cannot be exactly identical, we observe that the error type distribution of Kor-Native aligns with that of Native Korean (Shin et al., 2015) in that they are dominated by word spacing errors, which means that the types of errors of Kor-Native can serve as a reasonable representative to real-world writing errors made by Native Korean. After the filtering process described in Appendix A.2, we have 17,559 sentence pairs containing grammatical errors.

---

[5]http://kornorms.korean.go.kr/regltn/regltnView.do

[6]The merging codes are also open-sourced at our repository.

	Valid loss	Self-GLEU	GLEU on KoBART	Dataset size
Lang8 (Bef.)	1.53	15.01	19.69	204,130
Kor-Lang8(Aft.)	**0.83**	**19.38**	**28.57**	109,559

Table 2: Evaluation scores on the validation set for Lang8 (Mizumoto et al., 2011), the original lang8 dataset filtered by unique pairs in Korean, and Kor-Lang8, which is after the refinement by §3.3.

```
Orig: 그는 방을 저장한 다음에 갔다 학교에 .
Corr: 그는 방을 정리한 다음에 학교에 갔다 .
KAGAS output:
S 그는 방을 저장한 다음에 갔다 학교에 .
A 2 3|||VERB|||정리한|||REQUIRED|||-NONE-|||0
A 4 6|||WO|||학교에 갔다|||REQUIRED|||-NONE-|||0

Orig: He to school went after saving his room .
Corr: He went to school after cleaning his room .
KAGAS output:
S He to school went after saving his room .
A 4 5|||VERB||cleaning|||REQUIRED|||-NONE-|||0
A 1 3|||WO|||went to school|||REQUIRED|||-NONE-|||0
```

Figure 2: An example of an M^2 file output by KAGAS. Translated into English. Note that "to school" is treated as one word for the translation example.

3.3 Lang-8 Korean Corpus (Kor-Lang8)

Lang-8[8] is one of the largest social platforms for language learners (Mizumoto et al., 2011). We extract Korean data from the NAIST Lang-8 Learner Corpora[9] by the language label, resulting in 21,779 Korean sentence pairs. However, some texts are answers to language-related questions rather than corrections. The texts inside the raw Lang-8 corpus is noisy and not all of them form pairs, as previous works with building Japanese corpus out of Lang-8 (Koyama et al., 2020) also pointed out. To build a GEC dataset with high proportion of grammatical edits, we filtered out sentence pairs with a set of cleanup rules regarding the Korean linguistics, which is described in Appendix A.3.

Comparison with original Lang8. To prove the increased quality of Kor-Lang8, we compare the model training results and error type distribution between the original Korean version of Lang8 and Kor-Lang8. We perform minimum pre-processing to the original Korean Lang8-data which discard texts that do not have pairs and preserve unique original-corrected sentence pairs to enable training and make a fair comparison with Kor-Lang8, leaving out 204,130 pairs. Table 2 shows that a model trained with Kor-Lang8 achieve better results with lower validation loss, higher self-GLEU scores (§5.1), and higher scores when trained with KoBART, showing that there are **fewer outliers** on Kor-Lang8. [10] Figure 1 shows the difference in error type distributions before and after Lang8 refinement.

4 KAGAS

We propose Korean Automatic Grammatical error Annotation System (KAGAS) that automatically aligns edits and annotate error types on parallel corpora that overcomes many disadvantages of handwritten annotations by human (Appendix B.2). Figure 2 shows the overview of KAGAS. As the scope of the system is to extract edits and annotate an error type to each edit, our system assumes the given corrected sentence is grammatically correct. Then, our system takes a pair of the original sentence and the corrected sentence as input and output aligned edits with error types. We further extend the usage of KAGAS to analyze the generated text of our baseline models by each error type in Table 7 at Section 6. In this section, we describe in detail about the construction and contributions of KAGAS, with human evaluation results.

4.1 Automatic Error Annotation for other languages

Creating a sufficient amount of human-annotated dataset for GEC on other languages is not trivial. To navigate this problem, there were attempts to adapt ERRANT (Bryant et al., 2017) onto languages other than English for error type annotation, such as on Czech (Náplava et al., 2022), Hindi (Sonawane et al., 2020), Russian (Katinskaia et al., 2022), German (Boyd, 2018), and Arabic (Belkebir and Habash, 2021), but no existing work has previously extended ERRANT onto Korean. When

Error Code	Description & Acceptance Rate (%)	Example		
INS	A word is inserted. 100.00% ± 0.00%P	Original:	고등학교 때 어긴 경험	
		Corrected:	고등학교 때 **규칙을** 어긴 경험	
		Translation:	Experience to break a **rule** in high school	
DEL	A word is deleted. 100.00% ± 0.00%P	Original:	전쟁 **끝** 직후 장군들은 사형을 선고 받았다.	
		Corrected:	전쟁 직후 장군들은 사형을 선고 받았다.	
		Translation:	After the war, the generals are sentenced to death.	
WS	Spacing between words is changed. 100.00% ± 0.00%P	Original:	**이웃은** 더러워요.	
		Corrected:	**이 웃은** 더러워요.	
		Translation:	This **cloth** is dirty.	
WO	The order of words is changed. 97.44% ± 3.51%P	Original:	저는 **더 한국어를** 배우고 싶어요.	
		Corrected:	저는 **한국어를 더** 배우고 싶어요.	
		Translation:	I want to learn **Korean further**.	
SPELL	Spelling error 97.44% ± 3.51%P	Original:	파티에서 우리는 춤을 **쳐요.**	
		Corrected:	파티에서 우리는 춤을 **춰요.**	
		Translation:	We **dance** at the party.	
PUNCT	Punctuation error 98.72% ± 2.50%P	Original:	1993년 **의** 겨울의 일이였다.	
		Corrected:	1993년 **,** 겨울의 일이였다.	
		Translation:	It was 1993**,** a happening in winter.	
SHORT	An edit that does not change the structure of morphemes. 73.08% ± 9.84%P	Original:	한국어는 저한테 너무 어려운 **언어이었어요.**	
		Corrected:	한국어는 저한테 너무 어려운 **언어였어요.**	
		Translation:	Korean **Language** was too difficult to me.	
VERB	An error on verb 79.49% ± 8.96%P	Original:	어제 친구에게 편지를 **쌌어요.**	
		Corrected:	어제 친구에게 편지를 **썼어요.**	
		Translation:	I **wrote** a letter to my friend yesterday.	
ADJ	An error on adjective 73.08% ± 9.84%P	Original:	**진한** 친구	
		Corrected:	**친한** 친구	
		Translation:	A **close** friend.	
NOUN	An error on noun 75.64% ± 9.53%P	Original:	나중에 기회가 있을 때 한국에 **유학려** 가고 싶습니다.	
		Corrected:	나중에 기회가 있을 때 한국에 **유학** 가고 싶습니다.	
		Translation:	I want to study **abroad** in Korea in the future.	
PART	An error on particle 97.44% ± 3.51%P	Original:	**하와이에서** 사는 우리 사촌	
		Corrected:	**하와이에** 사는 우리 사촌	
		Translation:	My cousin living **in Hawaii**	
END	An error on ending 87.18% ± 7.42%P	Original:	오래 **기다려요.**	
		Corrected:	오래 **기다렸어요.**	
		Translation:	I **waited** for a long time.	
MOD	An error on modifier 89.74% ± 6.73%P	Original:	점심이 **나무** 작은 나머지 배고팠어요.	
		Corrected:	점심이 **너무** 작은 나머지 배고팠어요.	
		Translation:	I was hungry because I had **such** a small launch.	
CONJ	An error on conjugation 43.59% ± 11.00%P	Original:	오늘은 머리를 **잘라에** 갔다.	
		Corrected:	오늘은 머리를 **자르러** 갔다.	
		Translation:	I went to a barber to get my hair **cut** today.	

Table 3: Full category of error types used in KAGAS. Middle column shows acceptance rates by each error type on human evaluation along with explanations. Rightmost column shows examples of each error types. Others are classified as UNK.

extending ERRANT onto other languages, necessary changes about the error types were made such as discarding unmatched error types by ERRANT and adding language-specific error types.[11]

4.2 Alignment Strategy

Before classifying error types, we need to find *where* the edits are from parallel text. We first conduct sentence-level alignment to define a "single edit". We use Damerau-Levenshtein distance (Felice et al., 2016) by the edit extraction repository[12]

to get edit pairs. Note that we apply different alignment strategy from ERRANT on the scope of a "single" edit. We use Korean-specific linguistic cost,[13] so that word pairs with lower POS cost and lower lemma cost are more likely to be aligned together. Also, we use custom merging rules to merge single word-level edits into WO and WS. Therefore, the number of total edits and average token length on edits, and the output M^2 file made from KAGAS differs from that of ERRANT, since an M^2 file consists of edit alignment and error type (Fig. 2). This would result in different M^2 scores when applied to model output evaluation.

---

[11]VERB:SVA was discarded and DIACR was added for Czech (Náplava et al., 2022) In a similar way, KAGAS made necessary changes on error types regarding the linguistic feature of Korean, such as discarding VERB:SVA and adding WS.

[12]https://github.com/chrisjbryant/edit-extraction

[13]Kkma POS tagger, Korean lemmatizer

4.3 Error types for Korean

We describe how we consider the unique linguistic characteristics of Korean (Appendix B.1), and define 14 error types (Table 3).

Classifying error types in morpheme-level As Korean is an agglutinative language, difference between original and corrected word is naturally defined in morpheme-level. For example, 학교 에 ('to school') in Table 2 is divided into two parts, 학교('school') + 에('to'), based on its roles in a word. If this word is corrected to 집에('to home'), we should treat this edit as NOUN (학교 -> 집), and if this word is corrected to 학교에서 ('at school'), we should classify this edit as PART (Particle, since -서 is added). We need to break down original and corrected word-level edits into morphemes and only look at *morpheme-level differences* between the two when classifying error types. When conducting morpheme-level alignment, we utilize morpheme-level Levenshtein distance for Korean.[14] Also, the POS tags of Korean are based on *morphemes* and not *words*, meaning that there can be multiple POS tags for one word. Apart from POS tagging, KAGAS also considers the composition of edits (e.g. SHORT). Please refer to Appendix B.4 for detailed examples.

No PREP, but PART In Korean, morpheme (not word) align with the meaning. Therefore, "학교 에"(To school)->"학교에"(To-school) is WS, and "혁고에"(To-sceol)->"학교에"(To-school) is SPELL, which is different. In similar vein, There is no PREP (Positioning. "to" before "school") in Korean. They rather view them as postpositional particle (Positioning "-에" after "학교")

On the motivation of selecting 14 Error Types According to previous work that categorizes Korean grammatical error by frequency (Shin, 2007), Korean error types are divided by (1) Sound, (2) Format, (3) Spacing, and (4) The rest, meaning that orthographical errors were highly frequent in Korean error types. Therefore, we designed error types to focus on capturing frequent orthographical errors such as WS, SPELL, along with syntax and morphological errors such as WO and SHORT. There are 9 most important categories of POS for Korean (noun, pronoun, numeral, verb, adjective, postposition, pre-noun, adverbs, interjection), and a single word is divided into substantives (mostly by nouns) and inflectional words.[15] Most inflectional words are irregular and prone to change in format, and detecting those are also important. Therefore, we added 6 error types that can cover the 9 types of POS for Korean except for the numeral part[16] (noun&pronoun to NOUN, verb to VERB, adjective to ADJ, postposition to PART, pre-noun&adverb to MOD, interjection to PUNCT) and 2 error types for inflectional words (CONJ, END (=suffix)), which can be classified by the POS tagger. The result of 14 error types, motivated by both Korean linguistic characteristic information in terms of linguistic typology and orthographical guidelines, contain all the crucial, frequent error types.

About INS/DEL edits. Since Korean is a discourse-oriented language, one can omit the subject or the object in a sentence depending on the previous context. These cases are classified as INS/DEL edits, which are grammatically correct. There are also cases of INS/DEL that edits unnecessary modifiers, which is also non-grammatical edits but rather variations to sentences. Previous works that applied ERRANT onto other languages also discard INS/DEL edits or treat them in a similar manner to our work. Works on Hindi (Sonawane et al., 2020) and Russian (Katinskaia et al., 2022) only classifies R: (Replacement). For Arabic (Belkebir and Habash, 2021), Insertion and Deletion are not classified further other than token-level and word-level INS/DEL. We believe that some INS/DEL edits contain meaningful grammatical errors. However, following previous reasons and given our situation that we unfortunately don't have enough resources to conduct human evaluation of the subgroups of INS/DEL, we believe that not dividing INS/DEL any further would have more gains than losses regarding the reliability of KAGAS. DEL/INS examples are at Appendix B.3.1, and more details about selecting the granularity of error types are at Appendix Section B.6.

Priority between Error Types Due to the nature of Korean language, multiple error types can be classified for a single edit. However, we decided to output a single representative error type for each edit (Appendix B.5) by defining the priority between them in order to make a determinis-

---

[16]Due to lack of available data for evaluation, they are left as unclassified.

Dataset	Coverage	Overall acceptance rate			
		Total	Evaluator 1	Evaluator 2	Evaluator 3
Kor-Learner	81.56%	87.34% ± 5.49%P	84.32% ± 10.70%P	87.81% ± 8.26%P	89.88% ± 7.44%P
Kor-Native	90.92%	93.93% ± 2.18%P	92.92% ± 4.14%P	93.99% ± 2.82%P	94.87% ± 3.60%P
Kor-Lang8	82.52%	87.06% ± 4.67%P	84.72% ± 8.88%P	86.98% ± 7.02%P	89.48% ± 6.97%P

Table 4: The coverage and the overall acceptance rate of KAGAS, which is a weighted sum of individual acceptance rates by error types on real dataset distributions.

Correlation (kappa) scores		Value	Reliability
Fleiss'		**0.4386**	moderate
Krippendorff'		**0.4392**	moderate
Cohen's (pairwise)	Ann.1&2	0.5976	moderate
	Ann.1&3	0.4426	moderate
	Ann.2&3	0.3566	fair
	Average	**0.4656**	moderate

Table 5: Inter-annotator agreement on 3 different correlation metrics between 3 annotators. Note that Cohen suggested the kappa result be interpreted as follows: <0 indicating no agreement, <.2 none to slight, <.4 as fair, <.6 as moderate, <.8 as substantial, and <1 as perfect agreement. The result suggest moderate agreement.

tic, reliable system with clear evidence(Appendix C.2). Detailed steps are as follows: (1) We first classify edits that won't overlap with one another (INS/DEL/WS/WO/PUNCT) according to the current error type definition. (2) After that, we prioritized classifying frequent formal and orthographical errors such as SPELL and SHORT than the rest, since those errors are highly frequent in Korean Grammar. (3) When there are single POS types for an edit, we return the error types according to the POS. (4) When there were multiple POS types per an edit, we first check whether the edit was CONJ (a combination of VERB+ENDING or ADJECTIVE+ENDING). Others are left as unclassified. We detect INS and DEL directly by the outputs of sentence-level alignment. We merge the edits for WS and WO based on the syntactical appearance of the edits. For SPELL, we use a Korean spellchecker dictionary.[17] We utilize the Korean POS tagger (Appendix C.1) to classify other POS-related error types.

4.4 Evaluation of the Annotation System

We evaluate our system by 3 Korean GEC experts majoring in Korean linguistics (Table 4). First, we evaluate the acceptance rate of each error type by randomly sampling 26 parallel sentences with a single edit for each error type from our datasets.

One half (13 sentences) is written by native Korean speakers, and the other is written by KFL learners. In total, there are 364 parallel sentences in a random order. Each evaluator evaluated "good" or "bad" for each parallel sentences. The acceptance rate is the rate of "good" responses out of "good" and "bad" responses[18]. The overall acceptance rate is the sum of the acceptance rate of each error type weighted by the proportion of the error type in each dataset. Therefore, it depends on the distribution of error types in a dataset. By looking at the overall acceptance rate (Table 4), we can estimate that about 90% of the classified edits are evaluated as good for KAGAS on real dataset distributions. The coverage in a dataset is the rate of edits which is not identified as UNK (Unclassified). At Table 5, we can see that the inter-annotator agreements are moderate, meaning that the evaluation results are consistent between annotators to be reliable enough. It is also meaningful to note that, KAGAS has a very high human acceptance rate (>96.15%) for frequently observed error types on our dataset, such as WO, SPELL, PUNCT, and PART (PARTICLE). High acceptance rate for PART is especially meaningful since PART plays an important role in representing grammatical case (-격) in Korean. Detailed analysis including the evaluation interface is at Appendix C.3.1.

4.5 Contributions of KAGAS

To summarize, KAGAS is different from previous work such as (1) integration of morpheme-level POS tags, (2) using morpheme-level alignment strategy, and (3) Defining Korean-specific 14 error types. Following these reasons, we believe that KAGAS capture a more diverse and accurate set of Korean error types than a simple adaptation from automatic error type systems such as Choshen and Abend (2018) or ERRANT.

---

[17]https://github.com/spellcheck-ko/hunspell-dict-ko

| | Kor-Learner | | | | Kor-Native | | | | Kor-Lang8 | | | | Kor-Union | | | | Gen. time |
| | GLEU | M^2 | | | GLEU | M^2 | | | GLEU | M^2 | | | GLEU | M^2 | | | |
		Pre.	Rec.	$F_{0.5}$		Pre.	Rec.	$F_{0.5}$		Pre.	Rec.	$F_{0.5}$		Pre.	Rec.	$F_{0.5}$	
Self-Scores	25.54	1	0	0	25.71	1	0	0	20.01	1	0	0	21.66	1	0	0	
Hanspell	30.36	29.45	5.33	15.46	57.08	81.93	47.36	71.50	22.94	29.18	8.74	19.88	28.82	37.34	11.58	25.85	189.69
KoBART	**45.06**	43.35	**24.54**	37.58	**67.24**	75.34	**55.95**	70.45	28.48	**37.56**	11.62	25.93	**33.70**	**44.75**	**14.64**	**31.70**	38.25
KoBART + Kor-Union	42.66	**53.51**	21.18	**41.00**	59.71	**85.47**	47.38	**73.63**	**28.65**	37.46	**12.00**	**26.78**					37.51

Table 6: Full experiment statistics on the test set. KoBART outputs are averaged from outputs of 3 different seeds. Generation time (sentence/second) is measured by dividing the total number of sentences by total amount of generation time taken. Accuracies on the test set is reported with model checkpoints that has the highest GLEU validation accuracy.

5 Experiments

We conduct experiments to build an effective model to encourage future research on Korean GEC models. We report test accuracy using the model with the best validation GLEU score. Detailed experiment settings to reproduce our results appear in Appendix D.

5.1 Evaluation Metrics

We evaluate our model using M^2 scorer and GLEU (Section 2). Note that we can obtain M^2 scores as well as GLEU scores by making an M^2 file by KA-GAS. We also report self-scores (self-GLEU and self-M^2, obtained by treating original text as system outputs to each evaluation system) to compare the characteristics of the dataset itself. Higher self-scores would mean that corrected text is similar to the original text.

5.2 Baseline GEC system, Hanspell

Our primary aim is to build a first strong baseline model on Korean GEC. Therefore, we compare our methods with a commercial, well-known Korean GEC statistical system called Hanspell.[19] (Note that Hanspell is a completely different system from *Hunspell*.[20] a spellchecker) It is developed by the Pusan University since 1992 and it widely used in Korea since it is free and easily accessible through the web.[21]

5.3 Dataset split

We split datasets by the train_test_val_split function from sklearn.[22] The train, test, valid ratio is set to 70%, 15%, and 15% on seed 0. Then, we aggregate all three datasets to make *Kor-Union*. [23]

5.4 Model Training

We use the HuggingFace[24] implementation of BART by loading the weights from the pre-trained Korean BART (KoBART). We train models with multiple scenarios: (1) fine-tuning KoBART with 3 individual datasets, and (2) fine-tuning with Kor-Union and additionally fine-tuning on top of it with 3 individual data. We run each model with three different seeds and report the average score. For (1), we use a learning rate of 3e-5 for 10 epochs with a batch size of 64 for all datasets on a TESLA V100 13GB GPU. Other hyperparameters are the same as KoBART configurations. For (2), we use a learning rate of 1e-5.

5.5 Tokenization

We utilize the character BPE (Sennrich et al., 2016) tokenizer from HuggingFace tokenizers library, as KoBART used. Due to the limitations of the tokenizer, the encoded then decoded version of the original raw text automatically removes spaces between a word and punctuation (Appendix D.2). Therefore, naive evaluation of the generated output (decoded by the tokenizer) with the M^2 file made by raw text output is not aligned well, resulting in bad accuracy. Since we thought measuring the performance of the model has higher priority than measuring the performance of the tokenizer, we use the decoded version of text to train and make M^2 files for evaluation.

6 Results and Discussion

Effectiveness of Neural models As we can see in Table 6, the model trained with our dataset outperform the current commercial GEC system (Hanspell) on all datasets. It is notable in that

---

[19]https://github.com/9beach/hanspell

[20]https://github.com/hunspell/hunspell

[21]https://speller.cs.pusan.ac.kr/

[22]https://scikit-learn.org/stable/

[23]For fair evaluation, all 3 datasets are evenly distributed for each train, test, and valid split of Kor-Union.

[24]https://huggingface.co/transformers

		FULL	STD	INS	DEL	WS	WO	SPELL	PUN.	SHO.	VERB	ADJ	NOUN	PART	END	MOD	CONJ	UNK
GLEU	Hanspell	28.82	9.19	**21.89**	13.57	**46.91**	7.51	31.73	16.81	31.06	19.94	18.31	19.67	17.14	19.17	17.41	19.79	**21.15**
> 20	KoBART	33.70	6.02	**23.67**	18.22	**34.83**	13.76	**36.10**	20.29	**33.42**	27.96	29.27	25.09	27.15	27.72	23.68	25.95	26.44
Prec.	Hanspell	37.34	12.22	37.49	33.51	**71.98**	25.00	**48.21**	30.61	**46.70**	30.32	28.42	29.94	24.37	27.12	31.41	32.10	34.07
> 40	KoBART	44.75	6.78	**40.58**	**45.64**	**63.26**	37.89	**52.16**	42.35	**58.52**	**46.45**	**49.16**	**43.76**	**49.35**	**47.51**	**46.67**	**41.30**	**42.69**
Recall	Hanspell	11.58	9.00	**10.10**	7.40	**39.30**	5.26	**21.25**	6.14	**15.48**	6.62	6.12	8.14	5.01	5.64	7.08	8.60	9.69
> 10	KoBART	14.64	4.09	9.47	**12.05**	**21.48**	9.04	**22.54**	10.06	**19.61**	**13.29**	**13.92**	**12.46**	**14.30**	**13.23**	**12.70**	**12.34**	**12.68**
$F_{0.5}$	Hanspell	25.85	12.58	24.31	19.64	**61.72**	14.29	**38.46**	17.03	**33.28**	17.67	16.43	19.50	13.75	15.40	18.61	20.76	22.66
> 30	KoBART	31.70	6.46	24.47	29.28	**45.48**	23.11	**41.30**	25.78	**41.84**	**30.98**	**32.61**	29.11	**33.10**	**31.28**	**30.38**	28.09	28.97

Table 7: GLEU and M^2 scores on the generation output on the test set of Hanspell and KoBART on Kor-Union. The scores are divided by all 14 + UNK error types. For convenience, scores higher than a certain threshold are highlighted. PUN. and SHO. is for PUNCT and SHORT, respectively. The standard deviation(STD) of Hanspell is higher than that of KoBART, meaning that **scores by KoBART are evenly distributed for all error types**, while scores by Hanspell are biased toward WS and SPELL.

Hanspell is currently known as the best performing system open-source system for correcting erroneous Korean sentences. The result implies that our dataset helps to build a better GEC system, and our that our model can serve as a reasonable baseline that shows the effectiveness of neural models against previous rule-based systems on GEC. Moreover, the generation speed of our neural models (KoBART) is about five times faster than Hanspell, showing the efficiency as well as performance.[25]

Analysis by Error Types Here, we demonstrate the usefulness of KAGAS, which enables us to conduct a detailed post-analysis of model output by measuring model performance on individual error types. Table 7 shows score distributions on individual error types for Hanspell and KoBART on Kor-Union. (Full scores are at Appendix D.4). Compared with Hanspell, KoBART trained with our dataset generally perform better regardless of error types. In contrast, Hanspell's performance is very biased towards SPELL and WS.

Kor-Native Note that the performance of Kor-Native is much higher than the other datasets. The error type distribution (Figure 1) for Kor-Native aligns with Shin et al. (2015) that more than half of the dataset is on WS for native Korean speakers, which is different from learner datasets which has a more diverse set of error types. Therefore, it is easier for the model to train on Kor-Native than on other datasets.

7 Conclusion

In this work, we (1) construct three parallel datasets of grammatically incorrect and corrected Korean sentence pairs for training Korean GEC systems: Kor-Lang8, Kor-Native, and Kor-Learner. Our datasets are complementary representing grammatical errors that generated by both native Korean speakers and KFL learners. (2) to train and evaluate models with these new datasets, we develop KAGAS, which considers the linguistic characteristic of Korean and automatically aligns and annotates edits between sentence pairs. (3) We show our experimental results based on a pre-trained KoBART model with fine-tuning on our datasets and compare them with a baseline system, Hanspell. We expect that our datasets, evaluation toolkit, and models will foster active future research on Korean GEC as well as a wide range of Korean NLP tasks. Future work includes on further refining our proposed method, KAGAS, by extending the coverage and making more accurate error type classification.

---

[25]Note that we used the *hanspell-cli github wrapper* and not the *browser* to compare inference speed, for fair comparison. https://github.com/9beach/hanspell

Limitations

Our automatic error type system, KAGAS, has room for improvements. Although we got high human acceptance rate for the error type classification results of KAGAS, Our coverage of error types is about 80% to 90% (Table 4). Currently, our system rely on the Kkma POS tagger for Korean. We believe that the improvement of a POS tagger will enable KAGAS to define a more detailed error type classification with high coverage and reliability. Also, there could be other ways (or more efficient ways) to define and classify Korean grammatical edits. However, we would like KAGAS to be viewed as the first step towards the effort of making an automatic annotation tool for Korean GEC, which, though not perfect, have meaningful contributions to the field in its current form.

Future Directions. Currently, the 14 error types of KAGAS is focused to be as specifc as possible, while respecting both statistical characteristics of Korean language and incorporation into a reliable, deterministic system with high agreement of human evaluation. However, definment of a more richer error type classfication system derived from KAGAS such as differentiating between the typographical and phonetic errors would be an important future direction for our research, as both are defined as SPELL errors on our current system. It would require solving additional challenges of accurately disambiguating a writer's intention behind errors on a grammatical aspect. Another possible future direction would be applying data augmentation techniques on our datasets to boost the size of the training examples and obtain evaluation metric accuracy gains.

Ethics Statement

We have conducted an IRB for KAGAS human evaluation.[26]

Acknowledgement

We would like to thank all the LKLab lab mates, and Prof. Jungyeul Park for helpful discussions. We would also like to thank Geunhoo Kim, Jaeyun Kim, and from Sumin Lim for helping the dataset collection process and making the primary version of this paper. Lastly, I would like to thank Jinwoo Kim for helping me write a better rebuttal, and all the anonymous reviewers who really helped the most to make the current version of the paper.

---

[26]Approval number: KH2021-020

References

Riadh Belkebir and Nizar Habash. 2021. Automatic error type annotation for Arabic. In *Proceedings of the 25th Conference on Computational Natural Language Learning*, pages 596–606, Online. Association for Computational Linguistics.

Emily M Bender. 2011. On achieving and evaluating language-independence in nlp. *Linguistic Issues in Language Technology*, 6(3):1–26.

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, Brussels, Belgium. Association for Computational Linguistics.

Christopher Bryant, Mariano Felice, Øistein E. Andersen, and Ted Briscoe. 2019. The bea-2019 shared task on grammatical error correction. In *Proceedings of the Fourteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 52–75, 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.

Leshem Choshen and Omri Abend. 2018. Automatic metric validation for grammatical error correction. *CoRR*, abs/1804.11225.

Bernard Comrie. 1989. *Language universals and linguistic typology: Syntax and morphology*. University of Chicago 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.

Markus Dickinson, Ross Israel, and Sun-Hee Lee. 2011. Developing methodology for Korean particle

error detection. In *Proceedings of the Sixth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 81–86, Portland, Oregon. Association for Computational Linguistics.

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

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 *International Conference on Natural Language Processing*, pages 478–490. Springer.

Adam Christian Haupt, Jonathan Alt, and Samuel Buttrey. 2017. Identifying students at risk in academics: Analysis of korean language academic attrition at the defense language institute foreign language center. *Journal of Defense Analytics and Logistics*.

Ross Israel, Markus Dickinson, and Sun-Hee Lee. 2013. Detecting and correcting learner Korean particle omission errors. In *Proceedings of the Sixth International Joint Conference on Natural Language Processing*, pages 1419–1427, Nagoya, Japan. Asian Federation of Natural Language Processing.

Anisia Katinskaia, Maria Lebedeva, Jue Hou, and Roman Yangarber. 2022. Semi-automatically annotated learner corpus for Russian. In *Proceedings of the Thirteenth Language Resources and Evaluation Conference*, pages 832–839, Marseille, France. European Language Resources Association.

Satoru Katsumata and Mamoru Komachi. 2020. Stronger baselines for grammatical error correction using a pretrained encoder-decoder model. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pages 827–832, Suzhou, China. Association for Computational Linguistics.

Gyeongyeol Kim. 2020. Foreign students' korean writing errors - studying methods of language. *The Journal of Language and Literature*, 82:363–389.

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

Inhye Lee. 2018. Effects of contact with korean popular culture on kfl learners' motivation. *The Korean Language in America*, 22(1):25–45.

Kwankyu Lee. 2014. *Hangeul Orthography Impact Assessment*. 11-1371028-000540-01. KCI.

Minwoo Lee. 2020. An analysis of korean learners' errors by proficiency: Focus on statistical analysis using the multinomial logistic regression model. *Journal of Korean Language Education*, 31(2):143–169.

Myunghoon Lee, Hyeonho Shin, Dabin Lee, and Sung-Pil Choi. 2021. Korean grammatical error correction based on transformer with copying mechanisms and grammatical noise implantation methods. *Sensors*.

Sun-Hee Lee, Markus Dickinson, and Ross Israel. 2012. Developing learner corpus annotation for Korean particle errors. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 129–133, Jeju, Republic of Korea. Association for Computational Linguistics.

Sun-Hee Lee and Jae-young Song. 2012. Annotating particle realization and ellipsis in korean. In *Proceedings of the Sixth Linguistic Annotation Workshop*, pages 175–183.

Vladimir I. Levenshtein. 1965. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet physics. Doklady*, 10:707–710.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. 2019. Bart: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *arXiv preprint arXiv:1910.13461*.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. 2020. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online. Association for Computational Linguistics.

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

Jared Lichtarge, Chris Alberti, Shankar Kumar, Noam Shazeer, Niki Parmar, and Simon Tong. 2019. Corpora generation for grammatical error correction. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 3291–3301, Minneapolis, Minnesota. Association for Computational Linguistics.

Jin Hong Min, Seong Jun Jung, Se Hee Jung, Seongmin Yang, Jun Sang Cho, and Sung Hwan Kim. 2020. Grammatical error correction models for korean language via pre-trained denoising. *Quantitative Bio-Science*, pages 17–24.

Tomoya Mizumoto, Mamoru Komachi, Masaaki Nagata, and Yuji Matsumoto. 2011. Mining revision log of language learning sns for automated japanese error correction of second language learners. In *Proceedings of 5th International Joint Conference on Natural Language Processing*, pages 147–155.

Yunju Nam, Jewook Yoo, and Upyong Hong. 2018. The influence of constituents' semantic properties on the word order preference in korean sentence production. *Language and Information*, 22(1).

Jakub Náplava, Milan Straka, Jana Straková, and Alexandr Rosen. 2022. Czech grammar error correction with a large and diverse corpus. *Transactions of the Association for Computational Linguistics*, 10:452–467.

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.

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.

Chanjun Park, Sungjin Park, and Heuiseok Lim. 2020. Self-supervised korean spelling correction via denoising transformer. *7th International Conference on Information, System, and Convergence Applications*.

Gaoqi Rao, Qi Gong, Baolin Zhang, and Endong Xun. 2018. Overview of NLPTEA-2018 share task Chinese grammatical error diagnosis. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 42–51, Melbourne, Australia. Association for Computational Linguistics.

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.

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

Hocheol Shin, Buyeon Kim, and Kyubum Lee. 2015. A study on the hangeul orthography error status. *Grammar Education*, 23:63–94.

Seoin Shin. 2007. Corpus-based study of word order variations in korean. In *Proceedings of the Corpus Linguistics Conference (CL2007)*, volume 2730. Citeseer.

Ho-Min Sohn. 2001. *The Korean language*. Cambridge University Press.

Ankur Sonawane, Sujeet Kumar Vishwakarma, Bhavana Srivastava, and Anil Kumar Singh. 2020. Generating inflectional errors for grammatical error correction in Hindi. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing: Student Research Workshop*, pages 165–171, Suzhou, China. Association for Computational Linguistics.

Jae Jung Song. 2006. *The Korean language: Structure, use and context*. Routledge.

Reut Tsarfaty, Djamé Seddah, Yoav Goldberg, Sandra Kübler, Yannick Versley, Marie Candito, Jennifer Foster, Ines Rehbein, and Lamia Tounsi. 2010. Statistical parsing of morphologically rich languages (spmrl) what, how and whither. In *Proceedings of the NAACL HLT 2010 First Workshop on Statistical Parsing of Morphologically-Rich Languages*, pages 1–12.

Clara Vania and Adam Lopez. 2017. From characters to words to in between: Do we capture morphology? *arXiv preprint arXiv:1704.08352*.

Shih-Hung Wu, Jun-Wei Wang, Liang-Pu Chen, and Ping-Che Yang. 2018. CYUT-III team Chinese grammatical error diagnosis system report in NLPTEA-2018 CGED shared task. In *Proceedings of the 5th Workshop on Natural Language Processing Techniques for Educational Applications*, pages 199–202, Melbourne, Australia. Association for Computational Linguistics.

Wei Zhao, Liang Wang, Kewei Shen, Ruoyu Jia, and Jingming Liu. 2019. Improving grammatical error correction via pre-training a copy-augmented architecture with unlabeled data. In *Proceedings of the 2019 Conference of the North American Chapter of*

the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers), pages 156–165, Minneapolis, Minnesota. Association for Computational Linguistics.

Appendix for
Standardizing Korean Grammatical Error Correction:
Datasets, Evaluation, and Models

A Detailed instruction of dataset pre-processing

A.1 General

In original-corrected pairs, there are cases where punctuation and words are in one word for origial-corrected edit pairs, such as: "갔어." -> "갔어!" Since we are doing a word-level alignment, it seems inappropirate to classify this whole edit as "PUNCT". Therefore, in order to correctly get error type distributions per our dataset, we process all of our dataset to add spaces between punctuations ("갔어." -> "갔어 .", "갔어!" -> "갔어 !"). After this, only punctuations can be left for alignment. Now the edit pairs from the previous example are transformed into "."->"!", which seems very appropriate as an edit that can be classified as "PUNCT".

A.2 Kor-Native

Collecting correct sentences. We collect grammatically correct sentences from two sources:

1. 7,481 sentences from online education materials for Korean learners published by the Center for Teaching and Learning for Korean.[27]

2. 4,182 example sentences in an electronic dictionary written by NIKL.[28]

We have granted to apply changes to the original dataset (Additionally make grammatically wrong sentences out of correct sentence) and redistribute these datasets, under the Korean Gong-Gong-Nuri-4 license.[29] This license states that anyone can use Kor-Native for non-commercial purposes under proper attribution of source.

Collecting transcribed sentences. We read the correct sentences using Google Text-to-Speech (TTS) to
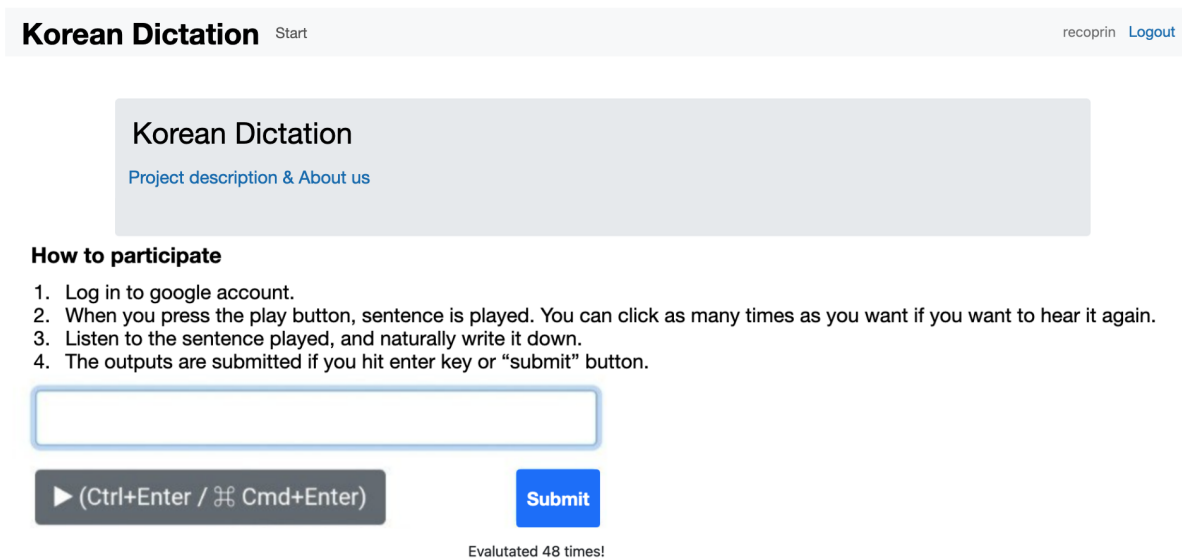
Figure 3: Demo page that we used for Kor-Native dataset collection. Translated into english.

native Korean speakers and let them dictate the sentences they hear on crowd-sourcing platforms. The demo page for the platform that we used is shown above.

We designed our method to deliver the correct sentences to the audience in oral because a written form may interfere the writing behavior of the audience. As a result, we collected 51,672 transcribed sentences.

---

[27] https://kcorpus.korean.go.kr/service/goErrorAnnotationSearch.do

[28] These sentences come from the National Institute of Korean Language which allows use of this corpus for research purposes.

[29] https://www.kogl.or.kr/info/license.do#04-tab

Filtering. Not all transcribed sentences contain grammatical errors. We filtered out transcribed sentences which do not contain a grammatical error by the following criteria:

1. If a correct sentence and its transcription are exactly same,

2. or if the differences between two sentences fall into any of the followings:

   - a punctuation,
   - related to a number,
   - a named entity.

A punctuation is not read by TTS. A number has multiple representations, e.g., "1" in Arabic numeral and "일" in Korean alphabet. Finally, we excluded transcribed sentences that are too short compared to the original correct sentence.

Why do native speakers mostly make spacing errors, while there are almost no errors on the learner corpus? Unlike English which generally requires a space between words, Korean often combine words without a space, depending on the context. The word spacing rule is very irregular with lots of exceptions. Consider the example 이웃은 (thiscloth) -> 이 웃은 (this cloth). In Korean, a sentence with incorrect word spacing is still comprehensible to some degree, thus people often don't strictly follow the word spacing rules. Incorrectly spaced sentences are accepted as long as they do not crucially affect readability, making word spacing rules even more difficult to memorize. On the other hand, Korean language learners may be more aware of accurate spacing due to their focus on language learning, and it is also likely that learners make other types of grammatical errors as frequent as spacing errors, which makes word spacing error much less dominant.

A.3 Kor-Lang8

We first extract incorrect-correct Korean dataset pair from the raw Lang-8 corpus. Then, we extract all pairs that contain Korean letters and preprocess the corpus to obtain (orignial, corrected) pairs. We apply various post-processing techniques into original raw Lang-8 corpora. Those techniques include: We discard pairs which:

- token length (when tokenized by kobart tokenizer) is longer than 200, since it consisted of meaningless repetition of words or numbers.

- contains language other than English, Korean, or punctuations, such as arabic or japanese characters.

- length of one token (splitted by space) is bigger than 20, since the sentences doesn't make sense by manual inspection.

- contains noisy words such as 'good', 'or', '/' inside.

- Doesn't consist of original <-> corrected pairs.

- length of each sentence is at least longer than 2. (naive length, not tokenized length)

We also compute the ratio (r_t) of the number of tokens of the post-edit to the pre-edit ($n_{t,pre}$, $n_{t,post}$). Similarly, we compute the ratio (r_l) of the lengths. Then, we retain the (pre-edit, post-edit) pairs satisfying the following conditions and discard the others: 1) $0.25 < r_t < 4$, 2) $0.5 < r_l < 1.25$, 3) $\min(n_{t,pre}, n_{t,post}) > 5$, and 4) the length of the longest common subsequence is greater than 10 characters.

Then, we modify each sequence by deleting the traces of unneeded, additional proof marks. Therefore we discard phrases which is inside brackets. Those indicate the subsequence SEQUENCES inside the text such as (SEQUENCES), {SEQUENCES}, <SEQUENCES>, or [SEQUENCES]. We discard them along WITH the brackets. In a similar context, there was multiple repetition of partiuclar tokens, such as " 안녕 홍대 !!!!!! ????". so we shortened repeated patterns and make it appear only once. Those special

tokens include [' ', '!', ';', '?', '◌̃', '>', '◌̂', '+', 'ㅠ', 'ㅜ', 'ㅋ']. After applying this, the original sentence is converted into "안녕 홍대 ! ?"

After this step, we filter sentences by leaving out only those whose jamo_levenshtein distance which is dicussed in Appendix A.3.1.[30] is smaller than 10. Pairs whose levenshtein distance is bigger than this threshold is likely to contain pairs that are not grammatical edits, but rather rephrases or additional explanations. Lastly, we retain pairs whose original and corrected pairs are unique and original and corrected sentences are not the same (there must be at least one edit).

After this step, there are 109,560 sentence pairs in this corpus. Full details about the modifying and filtering functions for lang8 are going to be opensourced, for reproducibility for everyone.

A.3.1 Jamo_levenshtein Distance

Levenshtein distance (Levenshtein, 1965) are computed between the pre-edit and the post-edit sentences. We compute the distance in morpheme-level and normalize it by the sentence lengths as follows:

$$||LD|| = \frac{LD(s_{pre}, s_{post})}{\min(|s_{pre}|, |s_{post}|) \log_{20} \min(|s_{pre}|, |s_{post}|)} \tag{1}$$

where s_{pre} and s_{post} denote pre-edit and post-edit, $|s|$ is length of sentence s, and $LD(\cdot, \cdot)$ is Levenshetein distance, and $min(\cdot, \cdot)$ is minimum value between two arguments. In other words, the jamo_Levenshetein Distance between pre-edit and post-edit is normalized by their sentence length of the shorter sentence, resulting in a smaller normalization effect for longer sentence (Grundkiewicz and Junczys-Dowmunt, 2014). We use an existing implementation[31] which is a function inside python library called soynlp.

A.4 Kor-Learner

The original corpus is a set of XML files with multiple tutors' tags and corrections to the errors of Korean learner essays.

A.4.1 Original XML format of the NIKL learner corpus

The NIKL learner corpus consists of correction edits classified with individual tags: some of them are: (1)The position of the error(morph from-to), (2)Morpheme-level suggestions(edits) to the error(Proofread), (3)The granularity level:whether it is replacement, insertion, deletion, and so on(ErrorPattern), (4)The level of the error(ErrorLevel), (5)The role of the error in a sentence(ErrorArea), or (6)Whether it is a written or spoken language. Below are examples of the original XML dataset:

```
Example Sentence of korean learner corpora (filetype: xml)
<s>오후 5시 반에 집에 들었어요.</s>
        ...
  <LearnerErrorAnnotations>
    <word>
     <w>들었어요.</w>
     <morph from="157" subsequence="1" to="162" wordStart="Start">
        <Proofread pos="VV">들어오</Proofread>
        <ErrorArea type="CVV" />
        <ErrorPattern type="REP" />
     </morph>
     <morph from="157" subsequence="2" to="162" wordStart="None">
        <Proofread pos="EP">았</Proofread>
     </morph>
     <morph from="157" subsequence="3" to="162" wordStart="None">
        <Preserved>어요</Preserved>
     </morph>
     <morph from="157" subsequence="4" to="162" wordStart="None">
        <Preserved>.</Preserved>
     </morph>
    </word>
  </LearnerErrorAnnotations>
```

The details of how we interpret and merge edits are explained at Appendix A.4.3.

A.4.2 Manual refinement step

As explained at Section 3.1, some XML files had empty edits, missing tags, and inconsistent edit correction tags depending on annotators. Of all the possible tags (Appendix A.4.1), it was common that not all ErrorArea, ErrorLevel, and ErrorPattern tags were present for each edit. Therefore we conduct a refinement step to ensure the quality of the dataset. We process the NIKL learner corpus by the following steps: First,

---

[30]It is a morpheme-level levenshtein distance for Korean.

[31]https://github.com/lovit/soynlp/blob/503eaee28799e9a3baf01483c6fc59e0db524fa3/soynlp/hangle/_distance.py

Merge all XML files into a single corpus. Then, we discard sentences with no or inconsistent proofread tags by *manual* inspection. For example, there were datasets labeled as "DELETE" for the proofread tags, where the place was originally meant to be the place for morpheme-level edits. We discard those datasets. Since the grammatical aspects of handling written and spoken languages are different, we discard datasets tagged as spoken language and leave only written language. Lastly, we validate the consistency of the types of errors tagged by different tutors and leave out only valid annotations. After this step, we build a corrected sentence from the original sentence and morpheme-level corrections by merging morpheme-level syllables into Korean words (Appendix A.4.3).

A.4.3 The merging process from Korean orthography guidelines

In order to build corrected word-level sentences by the NIKL learner corpus, we need to apply Korean orthography guidelines since the annotations are originally decomposed in morpheme-level. We explain in detail about the rules below:

- Section 18-6: When end of stem "ㅂ" is transformed to "ㅜ", write as transformed even it's against the guideline.[32]

- Section 34: When stem ends with "ㅏ, ㅓ", using '-아/-어, -았-/-었-' is harmonizing then write as it abbreviated.[33]

- Section 35: When stem ends with "ㅗ,ㅜ", '-아/-어, -았-/-었-' is harmonizing and abbreviated to "ㅘ/ㅝ,왔,웠", then write as it abbreviated.[34]

- Section 36: When "-어" is next to "ㅣ" and abbreviated to "ㅕ" then write as it abbreviated.[35]

We implemented the above Korean orthography guidelines and applied it to every sentence tokens gathered. These method provided sufficient coverage to handle all morpheme-level corrections of corpora. We will open-source the code. But there were exceptions and uncovered cases, so in case you want to build another corpora or utilize the code to merge morphemes into words, you may want to implement more Korean orthography guidelines on our code. We now take an example and show how we actually merged the morphemes. For example, the above XML file (Appendix A.4.1) contains correction annotations about a morphic change from "들었어요."(meaning : came back, mis-spelled) to "들어오"+"았"+"어요". In this case, stem "ㅗ" in "들어오" must be harmonized with ending "았" (by Korean orthography guideline, section 35). So "들어오"+"았" must be abbreviated to "들어왔". To handle these abbreviations, We followed these step:

- join all annotations.(i.e. "들어오"+"았"+"어요"+"." = "들어오았어요."

- decompose all tokens to syllables (i.e. "들" is decomposed to [ㄷ,ㅡ,ㄹ] and so on)

- if syllable sequence applicable to abbreviation rules, then merge.(i.e. decomposed syllable sequence (ㅇ,ㅗ,None),(ㅇ, ㅏ,ㅆ) comform with Section 35.)

- repeat until nothing to apply

B KAGAS Development Details

B.1 Brief introduction to Korean language

The current orthographical practice of Korean writing system, Hangul (한글), was established by the Korean Ministry of Education in 1988. One prominent feature of the practice is morphophonemic. This indicates that a symbol is the binding of letters consisting of morpheme-based syllables. For instance, though 자연어 in 'natural language' is pronounced as 자녀너 [tɕa.jʌ.nʌ], it should be written as 자연어

---

[32] https://kornorms.korean.go.kr/regltn/regltnView.do?regltn_code=0001&regltn_no=178#a238
[33] https://kornorms.korean.go.kr/regltn/regltnView.do?regltn_code=0001&regltn_no=178#a254
[34] https://kornorms.korean.go.kr/regltn/regltnView.do?regltn_code=0001&regltn_no=178#a255
[35] https://kornorms.korean.go.kr/regltn/regltnView.do?regltn_code=0001&regltn_no=178#a256

	Human annotations	KAGAS
Schema	Different by datasets. Cannot compare	Provides a unified schema
Decisions	Random Differs by annotators	Deterministic More trustworthy
Time/price cost	Experts needed. Expensive	No cost, can instantly get output

Table B.1: Benefits of KAGAS over human annotations.

since each of 자연 'natural', and 어 'language' is a morpheme with one or two syllables. Words, or Eojeol (어절) are formed by both content and functional morphemes in general. They are basic segments for word spacing in Korean. The rules for the word spacing are also described in the orthography guidelines, however, they are often regarded as complex ones for native Korean speakers (Lee, 2014).

In the view of linguistic typology, as mentioned, Korean is an agglutinative language in that each morpheme encodes a single feature. This turns out that the language has rich morphology such as various particles and complex conjugation forms. The example in (1) shows that each particles attached to a noun indicates a case marker such as nominative, accusative and the others. Furthermore, the affixes attached to a verb stem serve as functional morphemes pertaining to tense, aspect and mood. Another distinction of the language is that pro-drop or zero anaphora is abundant, which is common in morphologically rich languages (Tsarfaty et al., 2010). Particle omission is also frequent in colloquial speech (Lee and Song, 2012). These linguistic characterisitcs are different from the ones of fusional languages such as English and German where a concatenated morpheme has multiple features in usual (Comrie, 1989; Vania and Lopez, 2017).

(1)   수지-가   한나-에게   우체국-에서   편지-를   보내-는 중-이-라고
      Suzy-NOM Hannah-DAT post.office-LOC letter-ACC send-PRS currently.doing-ADJ-QUOT
      말-했-습-니-다
      say-PST-POL-IND-DECL
      'Suzy said that she was sending a letter to Hannah at the post office.'

The word order of Korean is relatively free. While the canonical word order of the language is Subject-Object-Verb (SOV), it is possible to change the positions of the words in a sentence. However, corpus and psycholinguistic studies have reported that the preferred order exists for adverbs with the conditions related to the meaning of the verb (Shin, 2007) and the specific types of the adverbs such as the time and place ones (Nam et al., 2018). Nevertheless, Korean speakers allow various word orders when comprehending and producing sentences in general.

B.2   Benefits of using KAGAS

Table B.1 shows the benefits of using an automatic system over human annotation. Making a fully human annotated dataset resource for Grammatical Error Correction is quite difficult and costly, and an automated version of it (KAGAS) could be a great alternative, and could even overcome many disadvantages of human annotation. Therefore, we emphasize here again the advantages of automatic error type correction system.

- KAGAS provides a unified schema for all Korean parallel datasets. In contrast, error types by human annotations are different by datasets, and thus hard to compare.

- KAGAS uses a deterministic, trustworthy decision on assigning error types, where it could be random or different by annotators for human annotations.

- KAGAS can be applied with no cost, and instantly get output while it takes a lot of time, money, and effort to hire experts for annotation and validate them. This particularly becomes a great advantage for datasets used for training neural models where the dataset size is often too large to conduct high-quality human annotation, and on other languages than English where experts are very expensive and difficult to hire.

B.3 More examples of Korean Error Types

B.3.1 INS & DEL

In Korean, one can omit the subject or the object in a sentence depending on the previous context, since it is a discourse-oriented language. In these cases, sentence with DEL and INS edits and also without DEL and INS edits are both grammatically correct. There were also cases of INS or DEL which are edits of unnecessary modifiers. This is also a case of non-grammatical edits but rather variations to sentences. For this reason, we felt no need to divide error types further for INS&DEL that mostly accounts for unnecessary, **non-grammatical edits**. Below are some INS & DEL examples from lang8.txt.

Line.1825 :
음악회에 가는 것은 좋아해서 ... (Like going to musicals)
INS->"나는" 음악회에 가는 것은 좋아해서 ... (I like going to musicals)
Line. 1909 :
날마다 "이" 일기에 쓰고 싶어요 (Want to write on this diary everyday)
DEL->날마다 일기에...(Want to write on diary everyday)
-All sentences are grammatically correct in Korean.

B.4 Detailed examples of Alignments and assignment of error types by KAGAS

We add some examples that describe how KAGAS assigns word-level POS error types by morpheme-level edits. Note that POS tagging is conducted by the kkma POS tagger.

- Word-level **ins**ertion

    - 음악회에 가는... => **나는** 음악회에 가는 ..
    - "나는" is inserted
    - INSERTION

- Morpheme-level **del**etion

    - 소풍(NNG) + 을(JKO) =>소풍(NNG)
    - 을(JKO) is deleted, thus labeled as PART(JKO is grouped to PART)

- Morpheme-level **ins**ertion

    - 유학(NNG) => 유학(NNG) + 러(NNP)
    - 러(NNP) inserted, thus labeled as Noun(NNP -> PART)

- Morpheme-level **sub**stitution

    - "싶습니다" => "싶어합니다"
    - tokenized by POS tagger as "싶"+"습니다" => "싶"+"어"+"하"+"ㅂ 니다"
    - 싶(VXA) + 습니다(EFN) -> 싶(VXA) + 어(ECD) + 하(VV) + ㅂ 니다(EFN)
    - (EFN->ENDING) to (ECD->ENDING, VV->VERB, EFN->ENDING)
    - (ENDING) -> (ENDING, VERB)
    - sum((ENDING), (ENDING,VERB)) -> (VERB, ENDING)
    - If we aggregate all POS included in this edit, we get (VERB, ENDING), and therefore is labeled into "CONJ".

Please refer to software:KAGAS/pos_granularity.py for full mapping of kkma POS tags grouped into our error types.

B.5 On assignment of single representative Error Types instead of multiple Error Types per a single edit

Defining error types that don't overlap with one another in the first place would be optimal, but unfortunately, defining meaningful error types for Korean that are mutually exclusive is almost infeasible. (Appendix A.4.1: The NIKL corpus tagged morpheme-level error types in 3 levels: the position of error types(오류 위치), ErrorPattern(오류 양상), and ErrorLevel(오류 층위).) Similarly, we want to clarify that the current implementation of KAGAS(software:KAGAS/scripts/align_text_korean.py#L404) has the ability to output all candidates of error type classifications(in formal aspect (INS/DEL/SUB), the POS of the edit, and the nature and scope of edit(SPELL/SHORT/The rest)). Currently, it is aggregated to a single error type, in the order of pre-defined priorities. While KAGAS can be easily extended to output multiple error types for a single edit, human evaluation and error type distribution analysis becomes much more complicated if we evaluate all possible error types per edit. For simplicity and clarity (and to make a deterministic reliable system), we decided to assign priorities and conduct human evaluation only on the highest priority error types. Please note that other works that extend ERRANT onto other languages also assign single error types to each edit (Náplava et al., 2022), (Sonawane et al., 2020), (Katinskaia et al., 2022).

B.6 The granularity of Error Types

Our primary goal on building KAGAS was to correctly classify error types in as much coverage as possible, while the human evaluation of KAGAS output is reliable enough. The first version of KAGAS was made after referring to the Korean orthography guidelines and other related work, and adjusting them into the ERRANT error types. It first contained a more diverse set of error types, with multiple error types assigned per an edit (e.g. SUB:VERB:FINAL_ENDING, SUB:VERB:DERIVATIONAL_ENDING, SUB:PARTICLE:OBJECTIVE, or INS:PUNCT). However, we noticed that there were 2 issues that prevented the practical and reliable use of the first version of KAGAS, and fixing these problems led to the current version described in the paper. First, the accuracy of kkma (POS tagger) was not good enough to ensure good quality of error types described previously in much detail, which is something that is beyond the scope of this work (We believe that the improvement of a POS tagger will enable KAGAS to define a more detailed error type classification with high reliability). Second, we could not perform reliable human evaluation with fine-grained error types. For reliable human evaluation we needed at least 26 samples per an error type - 13 in Kor-Native and 13 in Kor-Learner + Kor-Lang8 - to conduct a reliable human evaluation. Therefore, to ensure the quality of classification by KAGAS, error types without sufficient samples were aggregated into higher categories of similar groups or left as unclassified (at software:KAGAS/edit- extraction/pos_granularity.py).

C Implementation Details

C.1 Kkma POS Tagger

We use the Konlpy wrapper for Kkma Korean POS tagger,[36] to tag Part-Of-Speech information in a given sentence. We chose to use Kkma because it had the most diverse POS tags [37] among the konlpy POS taggers. However, Kkma fail to recover to the original form of a sentence after the output of POS tagging. Kkma outputs morpheme-level tags, and it erases whitespaces from the original input sentence. Therefore, recovering whitespaces after processing a sentence by Kkma is necessary, along with aggregating morpheme-level tags into word-level. We solve this issue by utilizing morpheme-level alignment for Korean.

C.2 Defining the priorities between error types

We wanted our system to be highly reliable and clear given the current available resources. Therefore, we prioritized classifying frequent, orthographic error types over POS classification.

After the output of the edit extraction, We use the allSplit method and merge mulitple edits as one

---

edit of word space and word order errors. For detection spell errors, we explained earlier that we use the Korean spellchecker dictionary.[38] Note that words that are proper noun is likely to be not included in the Korean dictionary, so spell errors are defined in a more narrower sense than it is currently thought of. We defined edits as spell errors only when original span wasn't inside the korean dictionary, but after editing, the edited word is inside the Korean dictionary. Therefore, corrections on proper nouns are treated as correct when there are classified as NOUN errors, not SPELL errors.

There were sometimes edits that could be both classified by one or more error types. For example, and insertion edit that added punctuation can be both classified as "INS" edits or "PUNCT" edits. In order to avoid this ambiguouity, we set the priority between edits. The priority is as follows.

- INS & DEL > the rest

- WS > WO > SPELL > SHORT > PUNCT > the rest

We informed this to participants for human evaluation to evaluate ambiguous edits on this priority. For Korean-specific linguistically aligned alignment, we computed similar with the English alignment system, but we defined Korean lemma cost using the soylemma's lemmatizer, and we defined the Korean content pos as NNG, NNP, NNB, NNM, NR, NP, VV, VA, VXV, VXA, VCP, VCN, MDT, MDN, MAG, MAC out of full pos tags for korean.[39]

C.3 Qualitative analysis on user evaluation.

C.3.1 Evaluation Interface

Figure 4 shows the evaluation demo interface that we used for human evaluation. We gave the full list of error types and made the evaluators to mark either 'good' or 'bad' about the error type classification.

C.3.2 About low-performing cases

Overall, the participants evaluated error types that could easily be identified by their forms with a higher proportion of 'good', and error types that relates to the POS tags as 'bad'. After manual inspection of edits that were classified as 'bad' by the Korean experts, we found that most of them were due to the limitations of the POS tagger. Most of the times the POS tagger fail to tag the correct POS for edit words, especially when there is a spelling error inside a word, or it is a pronoun. This explains why acceptance rates for POS-related error types had lower scores, for example, ADJ, NOUN, VERB, or CONJ. Also, after the main evaluation, we additionally asked the participants to classify edits that were marked as UNK, edits which KAGAS was unable to classify it to any error types. The participants classified most of the UNK edits as spelling errors. Since there are a lot of inflectional forms for a word for Korean, current dictionary-based spellchecker fail to identify all SPELL error edits. Therefore, we believe that KAGAS will benefit from the improvement of the Korean POS tagger and spell checker.

C.3.3 About selection of sentences for evaluation

For simplicity and clarity for annotators, we selected sentences with a **single** edit for each error type from our dataset for human evaluation. One concern could be that there could be a selection bias - straightforward cases could be selected for evaluation. We would like to first clarify that our 14 error types are entirely defined by local edits. In other words, the error type classification output of KAGAS is not affected by adjacent words or sentence structure (POS tagging is performed word-wise, and INS/DEL edits are not divided further). Therefore, we carefully argue that the validity of KAGAS is not affected by the number of edits and thus sentences with one edit can sufficiently represent the entire data, since the goal of human evaluation is to evaluate whether KAGAS correctly classifies word-level edits.

D Experimental Details

D.1 Details of Experimental Settings

We used a computational infrastructure which have 4 core CPU with 13GB memory and with one GPU(NVIDIA Tesla V100). All reported models are run on one GPU. We use the kobart pretrained model

---

[38]https://github.com/spellcheck-ko/hunspell-dict-ko
[39]http://kkma.snu.ac.kr/documents/index.jsp?doc=postag

Figure 4: Demo that we used for KAGAS system evaluation. Translated into english.

and kobart tokenizer. We allocate 70% of data set to train, 15% to test, and 15% to valid data sets by using Python scikit-learn library, sklearn.train_test_split function. GLEU(Napoles et al., 2015) scores are evaluated by the official github repository [40], and M^2 scores (Dahlmeier and Ng, 2012) are also evaluated on the official repository.[41]

D.2 Tokenizer issue on punctuation space recovery

Below is an example of the encoded and decoded outputs of the tokenizer.

```
>>> orig_text = "이게 뭔가요 왜 안돼요 ? ."
>>> orig_tokens = tokenizer.encode(orig_text)
>>> orig_tokens
[17032, 20156, 11900, 14851, 14105, ... 17546]
>>> decoded_text = tok.decode(orig_tokens)
>>> decoded_text
'이게 뭔가요 왜 안돼요?.'
>>> orig_text == decoded_text
False
```

We can see that spaces between punctuations and word disappeared from the decoded text, thus making it different from the original raw text. For this reason, we conduct the KAGAS experiments and report error

---

[40]https://github.com/cnap/gec-ranking
[41]https://github.com/nusnlp/m2scorer

	Kor-Learner					Kor-Native					Kor-Lang8					Kor-Union				
	GLEU	M^2			total time	GLEU	M^2			total time	GLEU	M^2			total time	GLEU	M^2			total time
		Pre.	Rec.	$F_{0.5}$			Pre.	Rec.	$F_{0.5}$			Pre.	Rec.	$F_{0.5}$			Pre.	Rec.	$F_{0.5}$	
Self-Scores	25.90	1	0	0	-	25.92	1	0	0	-	19.38	1	0	0	-	21.66	1	0	0	-
Hanspell	29.75	24.17	5.37	14.21	7	57.97	81.16	47.80	71.22	4	22.35	29.09	8.51	19.61	71	28.46	36.30	11.49	25.35	69
KoBART	46.94	43.95	26.35	38.76	1hr 4min	69.37	75.07	56.81	70.53	38min	28.57	37.69	12.64	26.96	3hr 39min	34.07	44.33	15.33	32.13	5hr 6min
KoBART + Kor-Union	44.66	51.94	23.55	41.83	1hr	61.64	83.95	48.55	73.25	39min	28.51	38.53	12.72	27.40	3hr 38min	-	-	-	-	-

Table D.1: We also report the evaluation results on valid sets which has the highest GLEU score. KoBART outputs are averaged from outputs of 3 different seeds. Here, we also report the total training time for 10 epochs, in the total time section.

	Kor-Learner	Kor-Native	Kor-Lang8
# r	.50	.46	.03
p	.06	.09	.92
Significant	O	O	X

Table D.2: Correlation between error type proportions(%) with respect to GLEU scores of KoBART + individual. The correlation is significant (p < .1) for Kor-Learner and Kor-Native.

type distributions on each datasets by raw text, but use the decoded version on evaluating model outputs. Full error type distributions on both raw text and decoded text is described on Table D.3.

BART. We use BART-base architecture, having the number of trainable parameters are 123M. we evaluate on the model which scores highest in GLUE scores. We first started with hyperparameters that were used to fine-tune BART on CNN-Dailymail task.[1] Among those parameters, we experimented on different dropout rates, and on the learning rates. We use the BartForConditionalGeneration structure by the huggingface library.[42] When generating, number of beams that we used for beam search is fixed to 4.

D.3 About pre-training with wikipedia dataset.

Since GEC suffers from lack of plentiful parallel data, we also tried to pre-train our model on Wikipedia edit pairs (Lichtarge et al., 2019) with a learning rate of 1e-05, and then fine-tune for 10 epochs on each individual datasets. However, we found out that KoBART is already a very strong pre-trained model, and the benefit from Wikipedia edit pair training is small. Therefore, we decided not to use the Korean Wikipedia edit pairs on our baseline experiment.

D.4 Further analysis on model outputs.

Accuracies improve linearly with the proportion of the training dataset. Kor-Native scores notably high on WS, while Kor-Learner scores poorly. According to Figure 1, Kor-Native has a large proportion of WS and Kor-Learner have only a small fraction. The same trend applies for PART on Kor-Learner datasets, compared with Kor-Lang8. Table D.2 is obtained by the indiviual error type proportion with respect to the GLEU scores shown in Figure 5. According to Table D.2, there is a positive correlation between the distribution of error types and the individual performance for Kor-Learner and Kor-Native. This means that when making a GEC model, training dataset distributions should differ in relation to what type of error types one wants to have high performance on. For example, if a model that performs well on ADJ errors is needed, Kor-Learner dataset should be utilized, and if a model that corrects WS errors very well is needed, the Kor-Native dataset should be used, and if one need to correct informal errors from KFL learners, using Kor-Lang8 would be the best, while using Kor-Native would be better to correct native speaker errors. Therefore, we believe all three datasets have their own purpose, and we provide them as separate three datasets without unifying them.

Comparison with KoBART and KoBART + Kor-Union The results on KoBART + Kor-Union is the results from the model fine-tuned twice, first with Kor-Union and then with the individual dataset. As we can see in Table 6, there is an improvement in precision and $F_{0.5}$ scores compared with KoBART+Kor-Union (KoBART fine-tuned on Kor-Union and then fine-tuned on each individual dataset again) than

---

[1]https://github.com/pytorch/fairseq/blob/master/examples/bart/README.summarization.md
[42]https://huggingface.co/transformers/
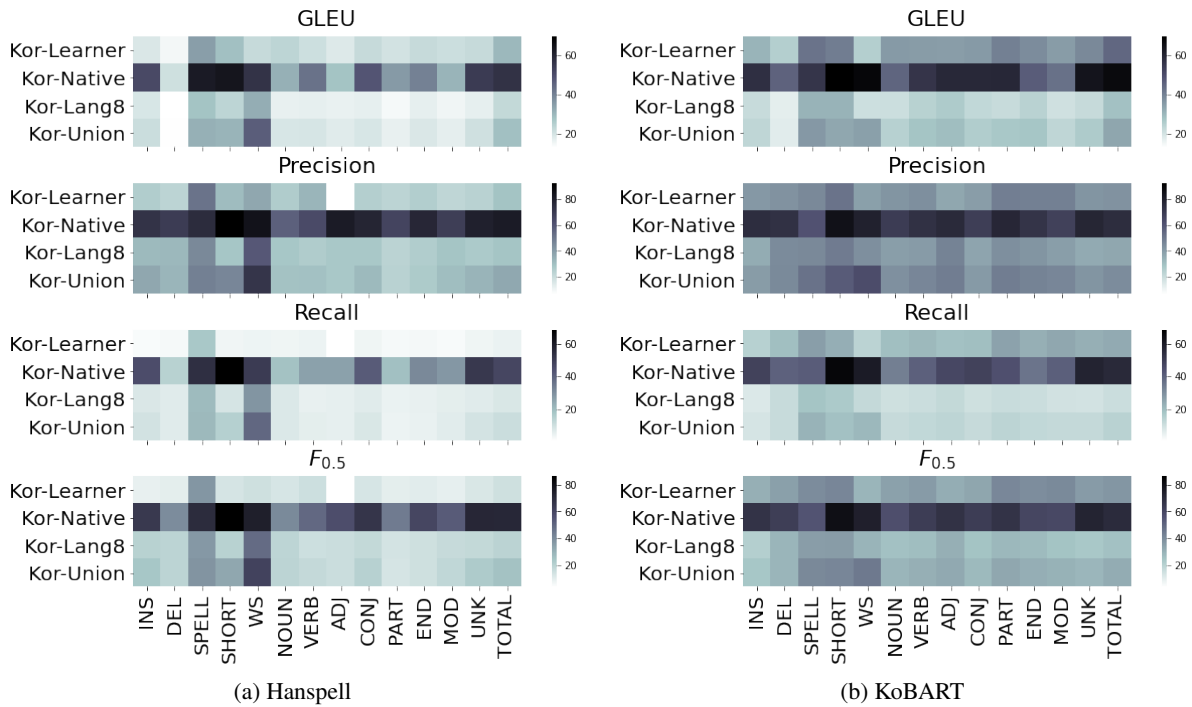
Figure 5: Heatmap illustration of generation output on test set of (a) Hanspell and (b) KoBART error types. We leave out WO and PUNCT due to the lack of examples in the test set. We can see that the scores of KoBART are similar over all error types, while Hanspell scores are biased toward word spacing (WS) and spelling (SPELL). Full values are in Appendix Table D.4.
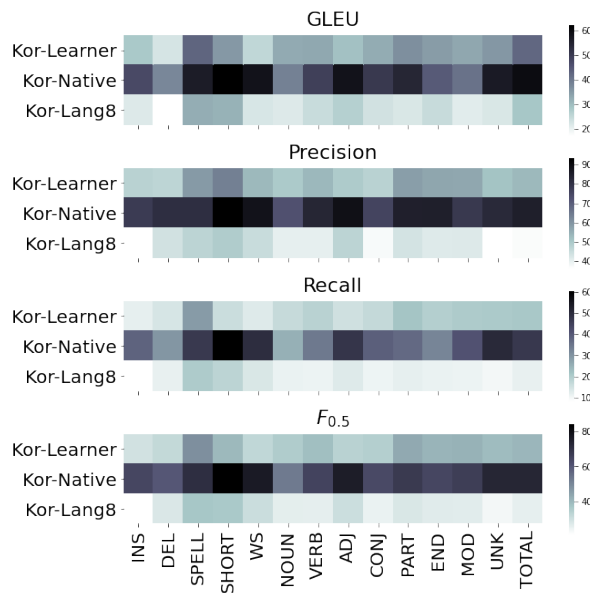
Figure 6: Heatmap illustration of error types generated by KoBART + Union and individual dataset fine-tuning. Note that the scoring range is different from that of Figure 7.

KoBART (fine-tuned directly on KoBART), meaning that all three datasets can help on improving the performance of the individual datasets. Analysis for KoBART + Kor-Union on each error type distributions shows similar trends with KoBART (Table D.4).

Additional Results   Figure 6 shows the test dataset heatmap for KoBART + Kor-Union. We can see that the trends are similar with that of KoBART. Also, Figure 7 shows the valid dataset heatmap of KoBART compared with Hanspell. The full error type scores are described in Table D.4. It first shows the count of occurences of the valid datasets used for generation and making heatmap illustrations. Also, it shows the
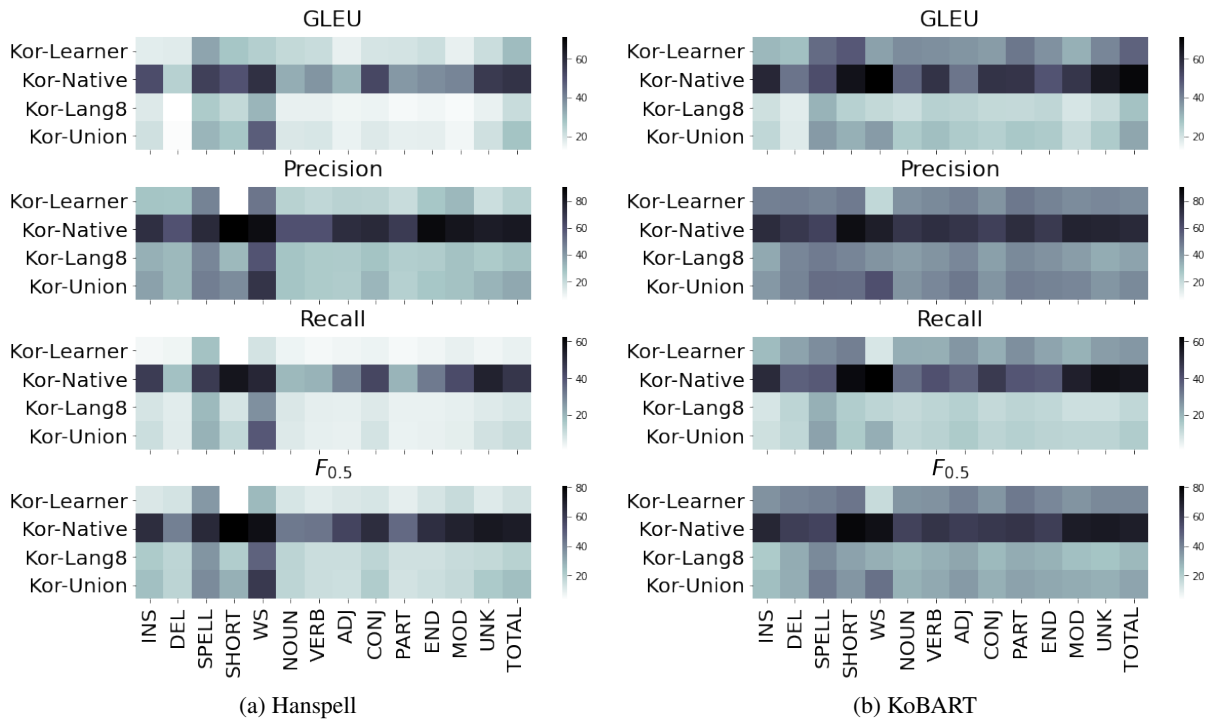
Figure 7: Heatmap illustration of generation output of valid set by (a) Hanspell and (b) KoBART error types. We can see that the distributions are similar to those on the test set.

full output of scores by all datasets with all methodologies.

Discussion on the ability of our model to refrain from editing when no fix is necessary   Another important aspect to grammatical error correction models would be about how these tools behave in the face of both grammatically incorrect and correct sentences. For this, we have evaluated the GEC model fine-tuned on the Kor-Native train dataset on both the source (grammatically incorrect) and target (grammatically correct) sentences of the 2,634 Kor-Native dev set. When we provided the model with grammatically correct sentences as input, the model output the exact same sentences as the original input sentence in 2184/2634 cases (82.92%). In contrast, when we provided grammatically incorrect sentences as input, the model only preserved the 400/2634 input sentences (15.19%) and fixed the input in 84.81% of the cases. This shows that while the model was only trained with grammatically incorrect sentences as input, it has developed an ability to determine whether a sentence is grammatically correct or not, thus refraining from editing in such cases. We will open-source the code to run this experiment and include the results on our next revision. Since we have presented our model as a baseline, we hope that many improvements can be made for this aspect in future work, e.g., explicitly being trained to preserve correct sentences.

Error Type(Full)	Kor-Learner		Kor-Native		Kor-Lang8	
	raw	decoded	raw	decoded	raw	decoded
INS	3352	3321	2004	1998	32665	28040
DEL	1652	1629	806	642	30666	27334
SPELL	6735	5642	3208	1078	19021	14506
PUNCT	0	19	2	79	284	3778
SHORT	363	374	115	277	857	974
WS	108	158	15625	15617	9766	9653
WO	0	0	45	45	701	676
NOUN	4879	5039	1442	1459	20300	20473
VERB	2456	2557	486	523	8616	8805
ADJ	411	444	55	88	1657	1753
CONJ	4917	5269	1699	2009	28775	31078
PART	16700	16692	1164	1195	39648	39175
END	7560	7310	591	683	25456	22495
MOD	1035	1043	238	258	5320	5260
UNK	9251	9940	2495	3921	39101	43132
TOTAL	59419	59437	29975	29872	262833	257132

Table D.3: Total count of edits on each error type on full data. Raw count is derived from KAGAS with raw dataset pairs, and Tokenized count is derived from KAGAS with tokenized->detokenized (decoded) dataset pairs using the Korean character BPE tokenizer.

			INS	DEL	SPELL	PUNCT	SHORT	WS	WO	NOUN	VERB	ADJ	CONJ	PART	END	MOD	UNK	TOTAL
Dataset Count	valid	Kor-Learner	450	249	828	0	65	19	0	785	394	77	769	2464	1084	149	1483	8816
		Kor-Native	331	94	196	21	34	2336	4	218	74	14	312	198	92	50	633	4607
		Kor-Lang8	4179	4131	2092	604	160	1434	95	3067	1350	248	4863	5878	3367	792	6486	38746
		Kor-Union	4960	4474	3116	625	259	3789	99	4070	1818	339	5944	8540	4543	991	8602	52169
	test	Kor-Learner	517	205	879	3	54	30	0	797	408	63	744	2405	1104	163	1506	8878
		Kor-Native	304	108	156	12	38	2383	10	208	90	15	308	176	103	40	578	4529
		Kor-Lang8	4187	4164	2151	571	136	1440	95	3119	1323	225	4587	6042	3283	815	6521	38659
		Kor-Union	5008	4477	3186	586	228	3853	105	4124	1821	303	5639	8623	4490	1018	8605	52066
Hanspell (valid)	Kor-Learner	gleu	18.45	18.50	34.48	0.00	27.67	25.77	0.00	23.23	22.42	17.02	20.26	20.54	21.78	17.08	22.20	29.75
		prec	28.42	27.78	46.34	100.00	6.67	50.00	100.00	23.54	21.74	23.68	22.99	19.75	28.11	31.94	19.45	24.17
		rec	3.71	4.36	17.82	100.00	1.42	9.30	100.00	4.55	3.58	4.09	4.81	3.18	4.29	5.60	4.29	5.37
		$f_{0.5}$	12.18	13.39	35.10	100.00	3.83	26.67	100.00	12.84	10.79	12.10	13.09	9.68	13.31	16.45	11.40	14.21
	Kor-Native	gleu	51.37	25.25	54.56	20.14	50.00	58.68	0.00	32.84	37.43	31.26	52.73	36.67	38.76	40.29	56.27	57.97
		prec	72.78	60.00	74.44	55.00	90.48	85.30	100.00	60.12	60.34	73.33	74.94	68.18	86.76	82.46	79.77	81.16
		rec	46.16	18.18	46.64	16.18	56.72	52.00	16.67	19.80	20.83	30.56	43.35	20.59	32.07	41.96	53.28	47.80
		$f_{0.5}$	65.25	41.10	66.51	37.16	80.85	75.61	50.00	42.72	43.75	57.29	65.40	46.63	64.69	69.12	72.56	71.22
	Kor-Lang8	gleu	18.93	13.22	27.21	15.59	23.51	32.10	10.02	16.90	17.91	15.54	16.44	14.69	15.46	13.74	17.12	22.35
		prec	33.97	32.04	46.06	29.82	34.62	58.39	22.73	27.45	25.93	22.52	29.70	24.80	26.66	26.03	26.88	29.09
		rec	8.91	6.89	20.20	5.88	8.70	27.86	4.17	7.48	5.91	4.76	7.95	5.35	5.39	5.20	7.38	8.51
		$f_{0.5}$	21.74	18.52	36.67	16.43	21.69	47.89	12.02	17.89	15.45	12.89	19.20	14.37	14.89	14.45	17.58	19.61
	Kor-Union	gleu	21.27	13.85	31.22	15.75	27.81	47.80	9.48	19.81	19.81	16.72	18.85	17.54	17.54	15.39	21.08	28.46
		prec	38.15	31.80	47.84	30.19	43.75	70.90	24.44	28.30	26.43	25.94	32.40	24.57	27.87	29.11	33.09	36.30
		rec	10.66	7.10	21.16	6.27	12.38	39.21	4.47	7.61	5.98	5.66	9.12	5.19	5.60	6.31	9.81	11.49
		$f_{0.5}$	25.16	18.76	38.21	17.12	29.03	61.04	12.91	18.34	15.70	15.12	21.45	14.07	15.52	16.90	22.44	25.35
Hanspell (test)	Kor-Learner	gleu	19.37	15.23	34.88	0.00	28.66	22.64	0.00	23.84	21.45	18.91	22.55	20.55	22.51	21.78	22.55	30.36
		prec	26.32	23.68	51.73	0.00	31.58	37.50	100.00	26.76	33.72	7.14	25.56	23.68	25.85	23.08	24.21	29.45
		rec	2.60	3.09	18.00	0.00	4.26	4.62	100.00	4.58	5.06	1.19	4.56	3.14	3.32	2.78	4.48	5.33
		$f_{0.5}$	9.32	10.16	37.63	0.00	13.82	15.46	100.00	13.59	15.80	3.57	13.30	10.26	10.97	9.39	12.87	15.46
	Kor-Native	gleu	51.18	21.25	62.94	12.02	64.19	56.69	12.02	31.62	43.16	28.11	48.57	35.82	40.25	31.07	54.57	57.08
		prec	72.97	69.23	75.38	88.89	92.31	85.28	50.00	57.06	63.86	81.82	78.01	66.06	77.50	68.42	79.82	81.93
		rec	45.49	14.95	54.35	15.38	68.57	50.52	11.11	19.28	26.63	26.47	41.89	19.67	31.96	28.57	51.58	47.36
		$f_{0.5}$	65.11	40.11	69.96	45.45	86.33	74.97	29.41	41.00	49.91	57.69	66.53	44.89	60.31	53.50	71.94	71.50
	Kor-Lang8	gleu	19.68	13.21	28.19	16.86	23.85	32.35	6.99	17.14	17.36	16.78	17.24	14.84	17.37	15.66	17.36	22.94
		prec	32.13	33.07	45.79	29.54	29.09	59.42	24.00	28.75	26.57	28.17	28.20	23.83	26.11	29.26	27.49	29.18
		rec	8.42	7.35	20.76	5.96	9.47	29.24	5.24	8.21	5.90	6.17	7.45	5.24	5.70	6.65	7.49	8.74
		$f_{0.5}$	20.56	19.45	36.90	16.49	20.57	49.25	13.99	19.16	15.62	16.45	18.11	13.94	15.21	17.41	17.92	19.88
	Kor-Union	gleu	21.89	13.57	31.73	16.81	31.06	46.91	7.51	19.67	19.94	18.31	19.79	17.14	19.17	17.41	21.15	28.82
		prec	37.49	33.51	48.21	29.54	46.70	71.98	25.00	29.94	30.32	28.42	30.12	24.37	27.12	31.39	34.07	37.34
		rec	10.10	7.40	21.25	6.14	15.48	39.30	5.26	8.14	6.62	6.12	8.60	5.01	5.64	7.08	9.69	11.58
		$f_{0.5}$	24.31	19.64	38.46	17.03	33.28	61.72	14.29	19.50	17.67	16.43	20.76	13.75	15.40	18.61	22.66	25.85
KoBART + finetune (valid)	Kor-Learner	gleu	30.81	29.15	45.31	0.00	48.83	35.07	0.00	38.95	38.51	37.23	35.85	42.99	37.91	32.06	40.01	46.94
		prec	47.19	47.74	46.12	100.00	48.60	21.57	100.00	42.50	44.37	46.92	41.95	49.38	46.45	43.99	45.07	43.95
		rec	19.11	24.09	28.46	100.00	31.16	8.53	100.00	21.98	21.58	26.79	21.58	28.12	23.92	21.07	25.18	26.35
		$f_{0.5}$	36.47	39.89	41.02	100.00	43.68	16.45	100.00	35.81	36.63	40.76	35.48	42.89	39.08	36.13	38.92	38.76
	Kor-Native	gleu	61.12	43.40	51.25	57.13	66.49	71.33	20.20	46.43	57.91	43.30	58.06	57.61	49.85	57.19	65.11	69.37
		prec	74.03	69.51	65.32	65.99	85.13	79.25	61.11	69.95	73.05	70.21	66.51	73.68	68.61	77.14	76.50	75.07
		rec	51.09	37.37	38.90	47.55	59.70	62.68	33.33	34.64	40.87	37.04	46.89	39.51	38.40	53.87	57.99	56.81
		$f_{0.5}$	67.92	59.31	57.50	61.24	78.44	75.27	52.19	58.08	63.09	59.52	61.36	62.78	59.28	70.99	71.90	70.53
	Kor-Lang8	gleu	21.36	18.22	31.65	21.81	25.35	23.19	16.61	21.89	25.20	24.62	23.19	22.90	23.82	20.10	22.50	28.57
		prec	35.88	45.60	48.41	37.63	45.81	41.98	38.18	39.35	41.99	45.16	38.59	44.46	42.35	39.19	35.20	37.69
		rec	8.70	12.97	21.48	9.36	15.25	13.14	10.17	11.91	13.04	14.44	11.93	13.20	12.53	10.36	10.43	12.64
		$f_{0.5}$	22.05	30.30	38.69	23.46	32.63	29.15	24.61	26.91	29.04	31.65	26.64	30.12	28.67	25.15	23.84	26.96
	Kor-Union	gleu	23.59	18.64	36.52	21.15	32.08	36.08	15.50	26.60	29.25	26.64	25.48	27.57	26.69	22.59	26.62	34.07
		prec	40.52	46.03	52.46	44.78	51.94	60.81	33.58	40.89	45.21	49.39	41.50	48.83	46.44	45.55	41.35	44.33
		rec	10.10	12.50	24.34	11.85	15.97	21.91	8.27	12.74	13.67	15.83	13.18	14.48	13.55	12.97	13.21	15.33
		$f_{0.5}$	25.27	29.92	42.60	28.77	35.74	44.79	20.83	28.72	30.90	34.67	29.00	32.71	31.24	30.27	28.97	32.13
KoBART + finetune (test)	Kor-Learner	gleu	31.05	25.37	42.99	0.00	41.12	25.49	0.00	35.83	35.38	35.25	35.78	40.53	38.22	35.44	38.96	45.06
		prec	41.60	43.17	45.89	0.00	51.96	39.43	100.00	42.56	44.24	37.54	40.11	48.88	48.18	48.20	42.49	43.35
		rec	15.23	20.10	27.00	0.00	23.64	14.36	100.00	19.89	21.36	19.45	19.89	26.32	23.56	25.29	23.44	24.54
		$f_{0.5}$	30.89	35.09	40.25	0.00	41.86	29.22	100.00	34.65	36.43	31.64	33.33	41.71	39.84	40.80	36.54	37.58
	Kor-Native	gleu	57.71	46.07	56.02	39.23	69.69	67.94	42.97	45.61	56.40	59.54	59.45	59.61	47.03	43.38	64.57	67.24
		prec	74.75	73.48	61.14	71.10	85.38	79.45	65.99	73.66	76.32	78.65	77.53	72.41	67.50	77.70	77.70	75.34
		rec	48.73	40.42	41.54	41.02	66.67	60.17	42.59	34.06	40.70	47.06	48.77	44.63	36.25	41.03	58.03	55.95
		$f_{0.5}$	67.53	63.13	55.84	61.97	80.81	74.65	59.37	57.32	63.39	67.81	63.49	67.54	60.33	59.78	72.77	70.45
	Kor-Lang8	gleu	22.29	17.64	31.13	20.61	31.14	21.49	12.42	24.87	26.36	23.31	22.39	26.36	23.31	22.55	22.55	28.48
		prec	36.23	45.31	45.98	39.63	49.45	44.27	29.64	39.95	41.03	47.55	38.21	44.18	42.71	40.07	36.67	37.56
		rec	8.12	12.14	18.77	9.13	17.26	12.73	6.99	10.85	11.50	13.01	10.92	11.95	11.56	10.20	10.07	11.62
		$f_{0.5}$	21.36	29.25	35.61	23.72	35.96	29.58	17.96	25.07	27.05	30.01	25.43	28.64	27.72	22.39	24.62	25.93
	Kor-Union	gleu	23.67	18.22	36.10	20.29	33.42	34.83	13.76	25.09	27.96	29.27	25.95	27.15	27.72	23.68	26.44	33.70
		prec	40.58	45.64	52.16	42.35	58.52	63.26	37.89	43.76	46.45	49.16	41.30	49.35	47.51	46.67	42.69	44.75
		rec	9.47	12.05	22.54	10.06	19.61	21.48	9.04	12.46	13.29	13.92	12.34	14.30	13.23	13.23	12.68	14.64
		$f_{0.5}$	24.47	29.28	41.30	25.78	41.84	45.48	23.11	29.11	30.98	32.61	28.09	33.10	31.28	30.38	28.97	31.70
KoBART + union +finetune (valid)	Kor-Learner	gleu	27.37	28.19	45.76	0.00	41.25	35.35	0.00	38.06	35.58	36.79	33.46	38.87	35.67	28.91	37.38	44.66
		prec	50.01	55.82	57.04	100.00	55.76	41.75	100.00	52.56	49.49	58.63	49.25	55.87	55.75	54.11	50.97	51.94
		rec	14.93	20.69	30.22	100.00	22.22	13.18	100.00	21.13	18.92	26.03	20.07	24.04	21.42	20.27	22.40	23.55
		$f_{0.5}$	33.98	41.64	49.42	100.00	45.45	29.05	100.00	37.40	34.65	46.86	38.14	44.16	42.19	40.59	40.61	41.83
	Kor-Native	gleu	52.63	34.43	50.01	44.80	50.88	62.02	19.93	39.88	51.19	41.17	49.43	51.92	44.72	53.01	58.43	61.64
		prec	76.54	74.25	75.03	69.65	90.85	88.28	100.00	73.44	83.64	74.32	75.39	77.49	83.49	92.65	80.50	83.95
		rec	42.21	25.88	45.99	41.02	49.26	52.94	33.33	35.25	35.32	33.33	39.61	32.57	34.78	41.52	51.52	48.55
		$f_{0.5}$	65.79	54.05	63.71	61.39	77.68	77.88	71.43	53.19	65.61	59.49	63.84	60.73	65.22	78.16	72.35	73.25
	Kor-Lang8	gleu	21.40	17.87	32.12	19.70	23.76	24.02	15.11	22.28	25.37	25.51	22.91	22.76	22.95	20.73	22.31	28.51
		prec	36.78	45.27	50.17	39.98	44.67	44.92	37.59	40.16	42.00	46.86	38.37	44.85	42.43	42.98	35.47	38.53
		rec	8.79	12.58	22.38	9.97	13.71	14.50	8.67	12.08	12.49	14.83	11.55	12.81	11.86	12.05	10.45	12.72
		$f_{0.5}$	22.46	29.79	40.19	24.94	30.86	31.53	22.54	27.41	28.52	32.72	26.20	29.89	28.00	28.39	23.98	27.40
KoBART + union +finetune (test)	Kor-Learner	gleu	28.45	22.88	43.09	0.00	35.61	25.51	0.00	32.85	33.28	29.59	32.73	37.01	34.93	33.36	35.74	42.66
		prec	47.90	47.19	59.30	0.00	64.04	53.56	100.00	50.13	53.62	49.76	47.99	58.27	57.13	56.74	50.49	53.51
		rec	12.18	14.61	29.07	0.00	16.31	13.33	100.00	16.98	18.80	15.48	17.25	21.82	19.64	20.49	20.83	21.18
		$f_{0.5}$	30.19	32.62	49.08	0.00	40.36	33.21	100.00	36.05	39.12	34.44	35.37	43.67	41.34	41.91	39.80	41.00
	Kor-Native	gleu	47.54	37.94	56.58	30.31	62.21	58.44	32.32	38.89	49.33	58.46	50.69	54.55	44.46	41.05	56.89	59.71
		prec	78.05	81.52	81.58	77.16	93.37	88.67	69.84	73.00	83.80	89.15	76.02	85.26	85.53	78.91	82.82	85.47
		rec	38.47	29.79	47.22	39.10	60.48	49.78	29.63	25.45	34.67	48.04	39.16	37.43	32.64	41.39	50.80	47.38

Kor-Lang8	$f_{0.5}$	64.68	60.50	71.15	64.56	84.17	76.68	54.75	53.14	65.28	76.04	63.95	67.89	64.57	66.77	73.54	73.63
	gleu	21.94	17.43	32.56	20.44	31.91	22.52	12.10	21.92	24.55	26.88	23.48	22.35	24.73	21.33	22.43	28.65
	prec	36.55	44.07	47.59	40.51	49.46	45.42	28.62	40.61	40.65	47.45	38.01	43.70	42.04	42.22	36.49	37.46
	rec	8.35	11.89	20.52	10.10	18.44	14.24	7.28	11.47	11.25	13.27	11.20	12.10	11.49	11.31	10.22	12.00
	$f_{0.5}$	21.81	28.56	37.64	25.26	36.93	31.55	18.00	26.90	26.68	31.28	25.68	28.68	27.42	27.26	24.08	26.28

Table D.4: Full scores on all error types, all datasets, on all methods, including valid dataset and test dataset. We also provide individual dataset count by error types on the top.

A For every submission:

☑ A1. Did you describe the limitations of your work?
On the limitations section, after the main content.

☒ A2. Did you discuss any potential risks of your work?
The dataset is available only for non-commercial research purposes.

☑ A3. Do the abstract and introduction summarize the paper's main claims?
Section 7 - conclusion

☒ A4. Have you used AI writing assistants when working on this paper?
Left blank.

B ☑ Did you use or create scientific artifacts?

Section 3

☑ B1. Did you cite the creators of artifacts you used?
Section 3

☑ B2. Did you discuss the license or terms for use and / or distribution of any artifacts?
Section 3

☑ B3. Did you discuss if your use of existing artifact(s) was consistent with their intended use, provided that it was specified? For the artifacts you create, do you specify intended use and whether that is compatible with the original access conditions (in particular, derivatives of data accessed for research purposes should not be used outside of research contexts)?
Section 3

☑ B4. Did you discuss the steps taken to check whether the data that was collected / used contains any information that names or uniquely identifies individual people or offensive content, and the steps taken to protect / anonymize it?
On section 3.2(Kor-Native), we have collected the transcribed dataset from the general public, and the user information is fully anonymized.

☑ B5. Did you provide documentation of the artifacts, e.g., coverage of domains, languages, and linguistic phenomena, demographic groups represented, etc.?
At the appendix, and at section 3.

☑ B6. Did you report relevant statistics like the number of examples, details of train / test / dev splits, etc. for the data that you used / created? Even for commonly-used benchmark datasets, include the number of examples in train / validation / test splits, as these provide necessary context for a reader to understand experimental results. For example, small differences in accuracy on large test sets may be significant, while on small test sets they may not be.
On section 4.3

C ☑ Did you run computational experiments?

Section 4

☑ C1. Did you report the number of parameters in the models used, the total computational budget (e.g., GPU hours), and computing infrastructure used?
At the appendix

The Responsible NLP Checklist used at ACL 2023 is adopted from NAACL 2022, with the addition of a question on AI writing assistance.

☑ C2. Did you discuss the experimental setup, including hyperparameter search and best-found hyperparameter values?
At the appendix and section 4

☑ C3. Did you report descriptive statistics about your results (e.g., error bars around results, summary statistics from sets of experiments), and is it transparent whether you are reporting the max, mean, etc. or just a single run?
At the appendix and section 4. At the appendix, we provide full results at the last part of the paper.

☑ C4. If you used existing packages (e.g., for preprocessing, for normalization, or for evaluation), did you report the implementation, model, and parameter settings used (e.g., NLTK, Spacy, ROUGE, etc.)?
Across sections, with footnote.

D   ☑ Did you use human annotators (e.g., crowdworkers) or research with human participants?
Discussed at section 4.4

☑ D1. Did you report the full text of instructions given to participants, including e.g., screenshots, disclaimers of any risks to participants or annotators, etc.?
It is illlustrated at appendix C.3, and at Figure 4 at the appendix.

☑ D2. Did you report information about how you recruited (e.g., crowdsourcing platform, students) and paid participants, and discuss if such payment is adequate given the participants' demographic (e.g., country of residence)?
Appendix C.3, and at section 2 and 3

☑ D3. Did you discuss whether and how consent was obtained from people whose data you're using/curating? For example, if you collected data via crowdsourcing, did your instructions to crowdworkers explain how the data would be used?
Appendix, and at section 2 and 3, and we conducted an IRB.

☑ D4. Was the data collection protocol approved (or determined exempt) by an ethics review board?
Yes, we conducted an IRB.

☑ D5. Did you report the basic demographic and geographic characteristics of the annotator population that is the source of the data?
The 3 human evaluators are Korean, majoring in Korean Linguistics. It is described at section 4.4 at the main paper.