Multilingual Grammatical Error Annotation: Combining Language-Agnostic Framework with Language-Specific Flexibility

Mengyang Qiu[1,2] **Tran Minh Nguyen**[2] **Zihao Huang**[2] **Zelong Li**[3] **Yang Gu**[2]
Qingyu Gao[2] **Siliang Liu**[2] **Jungyeul Park**[2,3]
[1]Trent University, Canada [2]Open Writing Evaluation, France
[3]University College London, UK [4]The University of British Columbia, Canada
http://open-writing-evaluation.github.io

Abstract

Grammatical Error Correction (GEC) relies on accurate error annotation and evaluation, yet existing frameworks, such as errant, face limitations when extended to typologically diverse languages. In this paper, we introduce a standardized, modular framework for multilingual grammatical error annotation. Our approach combines a language-agnostic foundation with structured language-specific extensions, enabling both consistency and flexibility across languages. We reimplement errant using stanza to support broader multilingual coverage, and demonstrate the framework's adaptability through applications to English, German, Czech, Korean, and Chinese, ranging from general-purpose annotation to more customized linguistic refinements. This work supports scalable and interpretable GEC annotation across languages and promotes more consistent evaluation in multilingual settings. The complete codebase and annotation tools can be accessed at https://github.com/open-writing-evaluation/jp_errant_bea.

1 Introduction

Grammatical Error Correction (GEC), which aims to automatically detect and correct errors in written text, has emerged as one of the most important and widely studied tasks in Natural Language Processing (NLP) for educational applications, particularly those supporting language learning and writing improvement. It benefits both native speakers (L1), by enhancing clarity and fluency in their writing, and non-native learners (L2), by providing immediate, structured feedback that reinforces correct grammatical patterns, boosts writing confidence, and, ultimately, supports language development and acquisition (Marjokorpi, 2023; Van Beuningen et al., 2012). Over the years, the lion's share of research has focused on advancing GEC systems—evolving from rule-based and statistical approaches to neural architectures, such as neural machine translation with transformers (Zhao et al., 2019) and, more recently, prompting-based approaches built on large language models (Zeng et al., 2024; for a comprehensive review, see Bryant et al., 2023).

Yet, automatic error annotation and evaluation play an equally critical role in GEC. Error annotation identifies and categorizes linguistic errors, while evaluation measures how effectively GEC systems correct them. Together, these two components help establish standardized benchmarks, influencing everything from system development to the quality of corrections eventually delivered to users. However, despite their importance, they have historically received less attention and are often treated as ancillary to system development and dataset creation.

Among existing tools for automatic error annotation and evaluation (e.g., M^2, Dahlmeier and Ng, 2012; GLEU, Napoles et al., 2015), errant (ERRor ANnotation Toolkit) has established itself as the *de facto* framework for English GEC. What makes errant stand out is its detailed linguistic annotations, with a total of 55 possible error types for English (Bryant et al., 2017). errant's significance was solidified in the *Building Educational Applications 2019 Shared Task: Grammatical Error Correction (BEA-2019)*, where it was used to standardize multiple datasets and served as the official scorer (Bryant et al., 2019).

While this toolkit has proven effective for English, further refinements are needed to improve its versatility and adaptability, especially in multilingual scenarios. Recent years have seen growing interest in multilingual GEC, as demonstrated by initiatives like the *MultiGEC-2025 Shared Task*, which brought together efforts across twelve typologically diverse European languages (Masciolini et al., 2025a,b). However, this surge in interest has outpaced the development of consistent multilingual annotation resources.

As noted by Masciolini et al. (2025a), only three languages in MultiGEC–namely Czech, German, and Greek–have received errant-style annotation. For the remaining languages, the authors acknowledge that, due to limited time and resources, they implemented only coarse-grained alignment between original and corrected texts to support holistic scoring, without access to the kind of detailed error analysis enabled by errant for English. Even in existing adaptations of errant for various languages, implementations vary considerably in their design choices–ranging from annotation label schemes to tokenization and part-of-speech (POS) tagging tools–and differ in the level of granularity applied to language-specific error types. Although differences in orthographies and morphosyntactic structures across languages are unavoidable, greater consistency in annotation practices is highly desirable.

To address these challenges, our goal is to develop a consistent and reusable framework for grammatical error annotation that can be readily adapted across typologically diverse languages. Drawing inspiration from the original errant's dataset-agnostic design, we extend its core philosophy to multilingual settings by separating the annotation pipeline into two components: a shared architecture that applies across languages, and optional extensions tailored to language-specific features. Even within the language-specific layer, we introduce structured templates for common error types, such as spelling, word order, and word boundary errors, which can be reused or adapted across languages with similar orthographic or syntactic patterns. In addition, our implementation relies on the stanza toolkit for tokenization and POS tagging, which provides standardized processing pipelines for over 70 languages (Qi et al., 2020), allowing our framework to be readily extended to annotate new GEC datasets of other languages when they become available.

The rest of the paper is organized as follows: §2 reviews the original errant framework, discusses challenges in its multilingual adaptations, and motivates the use of stanza for more consistent cross-linguistic preprocessing. §3 introduces our proposed grammatical error typology, which combines a language-agnostic core with structured, language-specific extensions. §4 presents our reimplementation of English errant and demonstrates the framework's applicability to multiple languages, ranging from generic use in European languages, to minor

template refinements for Korean, and deeper customization for Chinese. Finally, §5 summarizes our contributions and emphasizes the framework's flexibility and extensibility for multilingual GEC.

2 Background and Related Work

2.1 Description of errant

errant is a unified framework for error annotation and evaluation in English GEC. It provides a rule-based, dataset-agnostic approach for extracting and categorizing edits between original and corrected sentences, making it a crucial tool for system evaluation and benchmarking (Bryant et al., 2017).

At the core of its annotation pipeline is a linguistically enhanced alignment algorithm that identifies edit boundaries between sentence pairs. This algorithm, originally proposed by Felice et al. (2016), extends the Damerau-Levenshtein distance with a linguistically informed cost function that considers part-of-speech tags, lemmas, and character similarity. Unlike surface-level edit distance, this method prioritizes alignments between tokens that are syntactically or morphologically related (e.g., *meet* and *meeting*), and handles both one-to-one edits and multi-token reordering. A rule-based merging strategy is then applied to combine adjacent edits where appropriate, based on patterns frequently observed in learner data, such as phrasal verb edits. This alignment process significantly improves the consistency and quality of extracted edits (Felice et al., 2016).

Following alignment, errant applies a rule-based annotation scheme to categorize edits into fine-grained grammatical error types, enabling both comprehensive feedback and error-type evaluation. Specifically, it defines 25 primary error types based on POS and morphological properties obtained from spaCy[1], and further classifies them into three edit operations: Missing, Unnecessary, and Replacement, resulting in a total of 55 possible error types (e.g., R:VERB:TENSE indicates a replacement error related to verb tense). To store annotations, errant generates output in M2 format, the standard representation for GEC annotations since its adoption in the *CoNLL-2013 Shared Task* (Ng et al., 2013). Each annotated sentence consists of the original tokenized text (denoted by an *S* line) followed by one or more error annotation lines (*A* lines). Each *A* line specifies the error span, the

---

[1] https://spacy.io

error type, the suggested correction, and additional metadata (see Figure 1 for an example in English).

```
S This are a sentence .
A 1 2|||R:VERB:SVA|||is|||-REQUIRED-|||NONE|||0
A 3 3|||M:ADJ|||good|||-REQUIRED-|||NONE|||0
```

Figure 1: Example of an annotated sentence in M2 format from *BEA-2019*.

With edits extracted and categorized in a standardized format, errant can then be used to systematically evaluate GEC system outputs against gold-standard references. It calculates precision and recall between system-generated edits and gold-standard corrections and utilizes a harmonic mean $F_{0.5}$ score, which weights precision twice as much as recall to prioritize accurate and contextually appropriate corrections over excessive edits. Thanks to its detailed annotation schema, errant supports multi-granularity evaluation–analyzing system effectiveness not only at the overall level but also across specific error types and edit operations, enabling a fine-grained and transparent assessment of GEC models.

In *BEA-2019*, errant was used to standardize multiple datasets, some of which were annotated using different error type frameworks, while others lacked annotations entirely. This allowed for error distribution comparisons across datasets that were previously hindered by these annotation discrepancies. In addition, errant facilitated multi-level system evaluation by supporting error-type analysis across 24 main categories for all 21 participating teams[2]. This enabled a detailed assessment of each system's strengths and weaknesses and made it easier to identify which error types were the most challenging to correct (Bryant et al., 2019).

While errant provides a linguistically informed foundation for GEC annotation and evaluation, it is not without limitations. One minor issue is its tendency to overuse the OTHER category (i.e., unspecified errors), leading to less precise error categorization. For instance, certain errors that could be classified as specific grammatical types (e.g., verb tense or prepositions) are instead grouped under OTHER (Korre and Pavlopoulos, 2020).

Another issue, as discussed in Wang et al. (2025), arises in end-to-end evaluation scenarios. errant assumes pre-defined sentence boundaries, and mis-

---

[2]errant defines 25 categories, including UNKnown (error detected but unable to be corrected; Bryant et al., 2017). In *BEA-2019*, this category was not included.

alignment can result in an inability to generate evaluation results between gold-standard references and system outputs. However, in real-world GEC applications, such as learner essays, inconsistencies in sentence segmentation are a common issue, often caused by differences in preprocessing steps. To address this, Wang et al. (2025) introduced joint-preprocessing errant, incorporating an alignment-based approach to detect and resolve segmentation discrepancies before evaluation.

2.2 Challenges in existing multilingual adaptations of errant

Given its demonstrated success in English, errant has been adapted to multiple languages, including Arabic (Belkebir and Habash, 2021), Chinese (Hinson et al., 2020; Zhang et al., 2022; Gu et al., 2025), Czech (Náplava et al., 2022), German (Boyd, 2018), Greek (Korre et al., 2021), Hindi (Sonawane et al., 2020), and Korean (Yoon et al., 2023). While these adaptations have enabled broader use of errant-style annotation, they also reveal several challenges that arise when extending the framework to languages with a range of orthographic and morphosyntactic characteristics.

Inconsistent annotation labels A minor issue in multilingual adaptations of errant is inconsistent annotation labels for similar error types. The original errant for English defines three edit operations: Missing, Unnecessary, and Replacement, while treating word order (WO) as a main error category, similar to NOUN or VERB errors. In errant_zh, an adaptation for Chinese, these operations were denoted as insertion, deletion, substitution, and transposition (Hinson et al., 2020). Meanwhile, ChERRANT, another Chinese adaptation, later revised them to Missing, Redundant, Substitute, and Word-order (Zhang et al., 2022). While these differences do not affect core functionality, the lack of consistent labeling across adaptations can create confusion. Nevertheless, this issue is relatively straightforward to address, as it primarily involves terminology standardization.

Inconsistent preprocessing tools A moderate challenge in multilingual GEC annotation lies in linguistic preprocessing tools, particularly word segmentation and POS tagging. While spaCy, the default NLP library in errant for English, supports multiple languages, its effectiveness varies across linguistic systems, prompting many adaptations to incorporate alternative tools. For exam-

ple, German `errant` retained much of the `spaCy` pipeline but found its lemmatization insufficient, replacing it with TreeTagger for better accuracy (Boyd, 2018). For non-European languages, entirely different tools are used, such as Kkma POS Tagger for Korean KAGAS (Yoon et al., 2023) and LTP (Language Technology Platform) for Chinese ChERRANT (Zhang et al., 2022).

Although these variations allow for language-specific optimizations, different tokenization strategies and POS tagging schemes can lead to discrepancies in how errors are identified and classified. This is particularly problematic for multilingual GEC models, where standardized evaluation across multiple languages is crucial. Since a system's measured performance is inherently tied to how its errors are annotated, such variations can obscure true system similarities or differences and compromise the reliability of multilingual benchmarks.

Inconsistent annotation granularity A more significant challenge in multilingual GEC annotation involves the varying levels of granularity for language-specific errors. While `errant` provides detailed error categories for English, adaptations to other languages, especially non-European languages, often fail to maintain this level of detail. For instance, `errant_zh` uses only four basic edit operations at the character level, without POS information (Hinson et al., 2020).

Recent work has begun addressing this limitation by introducing more fine-grained annotations tailored to specific linguistic properties. For example, Gu et al. (2025) propose a refined error typology for Chinese that accounts for phonetic similarity, visual similarity, and other structural errors specific to Chinese. While this framework was developed for Chinese, many of its principles can be readily applied to languages with similar logographic orthographies.

2.3 `stanza` as a multilingual alternative to `spaCy`

The original `errant` framework relies on `spaCy` for preprocessing tasks such as tokenization and POS tagging. However, `spaCy`'s multilingual capabilities are relatively limited, covering only a small number of languages and exhibiting inconsistent performance across linguistic families. This has contributed to the fragmented landscape of language-specific adaptations in prior `errant` variants.

To promote cross-lingual consistency, our implementation adopts `stanza` (Qi et al., 2020), a fully neural pipeline trained on Universal Dependencies (UD) and other multilingual corpora. `stanza` supports over 70 languages and applies a consistent architecture and UD-based annotation scheme across its modules—including tokenization, multi-word token expansion, POS and morphological tagging, dependency parsing, and named entity recognition[3]. Benchmark evaluations indicate strong performance across typologically diverse languages.

Crucially, our aim is not to promote a specific tool, but to align the preprocessing stage with the same linguistic principles that underlie our error typology. Like UD, our taxonomy adopts a cross-linguistically consistent core structure with optional language-specific extensions. Using a UD-compatible parser such as `stanza` ensures that all languages are analyzed under a shared morphosyntactic framework, which is essential for scalable and comparable multilingual grammatical error annotation. In this sense, it is the UD standard, rather than any particular NLP library, that provides the conceptual and practical foundation for our approach.

3 Multilingual Error Typology

An error typology provides a systematic framework for identifying, classifying, and analyzing errors in written text. We propose a two-tiered typology consisting of a language-agnostic foundation and a set of structured, language-specific extensions. The first level includes the widely adopted MRU (Missing, Replacement, Unnecessary) framework, ensuring consistency in annotation and evaluation across linguistic systems. The second level provides a structured template for language-specific extensions, allowing related languages to share annotation strategies and avoid redundant reimplementation. By designing this layered approach, we promote standardization across languages while allowing flexibility for language-specific refinements.

3.1 Language-agnostic error annotation

The MRU framework classifies errors into three core operations: Missing (M), where essential elements are omitted; Replacement (R), where an incorrect element substitutes the correct one; and Unnecessary (U), where superfluous elements cause redundancy.

---

[3]https://stanfordnlp.github.io/stanza/

Each error is further specified with POS tags for precise categorization.

Missing (M) An essential linguistic element is omitted from a sentence, leading to incomplete or ungrammatical structures. These errors typically involve the absence of words or phrases necessary for grammaticality or semantic clarity, such as missing determiners. In annotation, missing errors are further categorized based on POS tags or syntactic functions. For example, `M:NOUN` indicates a missing noun.

Replacement (R) An incorrect linguistic element is used in place of the correct one. These errors frequently involve incorrect word forms or inappropriate lexical choices (e.g., `R:VERB` denotes an erroneous verb substitution). To further reduce ambiguity in annotation, we implement the $R:P_1 \rightarrow P_2$ pattern, where P_1 is replaced by P_2.

Unnecessary (U) A superfluous linguistic element is present in a sentence, resulting in redundancy or ungrammaticality. These errors often involve extraneous words or phrases that disrupt sentence structure or meaning. Similar to missing and replacement errors, unnecessary errors are annotated with POS information to specify the redundant element. For example, `U:DET` denotes an unnecessary determiner.

3.2 Language-specific error annotation

To accommodate language-specific characteristics, we introduce a set of structured extensions to the `MRU` core. Our approach maintains consistency with established annotation schemes such as `errant` while capturing morphological and syntactic errors unique to different languages. Algorithm 1 presents our proposed classification routine for Replacement errors. Given a pair of word sequences—the source ($\mathcal{S}$) and the target ($\mathcal{T}$)—the algorithm classifies the error into one of the following types: spelling errors (R:SPELL), word order errors (R:WO), or word boundary errors (R:WB). Spelling similarity is computed using two metrics: phonetic similarity and visual (shape-based) similarity. The thresholds α_1 and α_2 govern sensitivity to phonetic and visual matches, respectively.

The classification uses the following notation:

- $\mathcal{S}, \mathcal{T}$: word sequences in the source and target sentences.

- SIM(*phonetic*) and SIM(*shape*): similarity functions comparing pronunciation and visual form.

- SET($\mathcal{S}$): returns a bag-of-words representation of $\mathcal{S}$, disregarding word order.

- MERGE($\mathcal{S}$): reconstructs a character sequence from the tokenized input (i.e., merging tokens without spaces) to test for boundary alignment.

This structured yet extensible framework allows consistent error categorization across languages, while also accommodating language-specific scripts and segmentation conventions.

Algorithm 1 Pseudo-code for error classification

1: **function** ERRORCLASSIFICATION ($\mathcal{S}, \mathcal{T}$):
2:    **if** (SIM (*phonetic*) $> \alpha_1$) $\wedge$ (SIM (*shape*) $> \alpha_2$) **then**
3:       **return** R:SPELL:PHONOGRAPHIC
4:    **else if** (SIM (*phonetic*) $> \alpha_1$) **then**
5:       **return** R:SPELL:PHONETIC
6:    **else if** (SIM (*shape*) $> \alpha_2$) **then**
7:       **return** R:SPELL:SHAPE
8:    **else if** (SET ($\mathcal{S}$) == SET ($\mathcal{T}$)) **then**
9:       **return** R:WO
10:   **else if** (MERGE($\mathcal{S}$) == MERGE($\mathcal{T}$)) **then**
11:      **return** R:WB
12:   **end if**
13:   **return** {R}

Spelling errors We classify spelling errors by their underlying cause: sound-based phonetic similarity (R:SPELL:PHONETIC), visual resemblance in orthographic shape (R:SPELL:SHAPE), or a combination of both (R:SPELL:PHONOGRAPHIC). For sound-based phonetic errors, we introduce a transcription system to represent pronunciation, such as a pronouncing dictionary for English, *pinyin* for Chinese, or romanization for other languages. This allows us to compare words based on their phonetic similarity and identify errors caused by mispronunciation or phoneme substitution. For orthographic shape errors, we assess visual similarity by converting characters into font images and applying similarity metrics. This approach helps detect errors caused by visually similar characters, such as mistyped letters in Latin-based scripts or miswritten strokes in logographic writing systems like Chinese and Japanese. By combining these methods, we systematically classify and analyze spelling errors across different languages.

Word order errors Word order errors are flagged when the source sequence ($\mathcal{S}$) and the target sequence ($\mathcal{T}$) contain the same set of words

but differ in arrangement. In such cases, all words from the original sequence are retained, but their relative positions are altered. These errors are particularly common in languages with flexible word order, where reordering affects grammaticality or readability. Identifying and categorizing such errors enables more structured syntactic analysis and improves grammatical error correction.

Word boundary errors Word boundary errors occur when the source sequence ($\mathcal{S}$) and the target sequence ($\mathcal{T}$) yield the same sequence after merging their respective word components. These errors typically involve incorrect spacing, where words that should remain separate are mistakenly merged, or conversely, a single word is improperly split into multiple tokens. Since the fundamental content remains unchanged but the segmentation differs, such errors impact readability, syntactic structure, and lexical integrity. Addressing these errors ensures accurate word segmentation and proper grammatical representation.

Figure 2 illustrates representative examples of the three major subtypes of Replacement errors classified by our algorithm.

R:SPELL:PHONETIC	*their → there*
R:WO	*You can help me → Can you help me*
R:WB	*ice cream → icecream*

Figure 2: Examples of Replacement error types: phonetic spelling error (R:SPELL:PHONETIC), word order error (R:WO), and word boundary error (R:WB).

4 Implementation of Multilingual Error Annotation

Our implementation demonstrates how grammatical error annotation can be consistently extended across typologically diverse languages. We begin by reimplementing errant for English using stanza and validating its performance. We then apply the same system to other European languages without language-specific modules. For Korean, we introduce targeted refinements using language-specific templates. Finally, for Chinese, we show how deeper customization can be incorporated by modifying segmentation and retraining the stanza pipeline.

4.1 Reimplementing errant for English

We reimplemented errant using stanza for POS tagging and dependency parsing, as described in §2.3. This enables our annotation system to be more consistent across languages while preserving the linguistic precision required for English-specific grammatical labels.

We integrated the English-specific classification module from the original errant, which identifies detailed grammatical error types, such as NOUN:POSS for possessive noun suffix errors. This module relies on universal POS tags (Petrov et al., 2012) and dependency relation tags to categorize errors. For instance, if the first token in an edit is tagged as PART and its dependency relation is case:poss, the classifier assigns the NOUN:POSS label accordingly.

A key distinction between the original errant and our implementation lies in error categorization. As illustrated in Figure 3, errant originally annotates *that is* as a missing OTHER error, whereas our implementation classifies it more precisely as a missing PRON error. Additionally, we refine verb annotation by distinguishing auxiliary verbs in passive constructions, categorizing *is played* as a Replacement error from VERB to AUX VERB. These refinements enhance interpretability by providing more specific and linguistically meaningful labels for complex constructions.

```
Original errant:
S Volleyball is a sport play every place ...
A 4 4|||M:OTHER|||that is|||REQUIRED|||-NONE-|||0
A 4 5|||R:VERB:FORM|||played|||REQUIRED|||-NONE-|||0

Our implementation:
S Volleyball is a sport play every place ...
A 4 4|||M:PRON|||that|||REQUIRED|||-NONE-|||0
A 4 5|||R:VERB → AUX VERB|||is played|||REQUIRED|||-NONE-|||0
```

Figure 3: Differences between errant and our implementation

To evaluate the overall alignment, we compared both implementations using outputs from the state-of-the-art GEC system, T5 (Rothe et al., 2021). The results in Table 1 support that our implementation reproduces errant's scores, with only minor variations.

	TP	FP	FN	Prec	Rec	$F_{0.5}$
errant	2589	1639	4030	0.6123	0.3911	0.5501
Ours	2565	1613	4028	0.6139	0.3890	0.5503

Table 1: GEC results for English using T5

4.2 Applying universal annotation to European languages

Without language-specific classification modules, our grammatical error annotation system remains capable of generating generic error annotations using the core MRU framework combined with POS

Czech	S Mám velkou rodinu , tak nemohla jsem mít naději , že něco dostanu .
Náplava et al. (2022)	A 5 7||R:WO||jsem nemohla||REQUIRED||-NONE-||0
Ours	A 5 7||R:VERB AUX -> AUX VERB||jsem nemohla||REQUIRED||-NONE-||0
	('I have a big family, so I couldn't hope to get anything.')
German	S Dagegen wieder , bekommen BA Studenten die ein extra Jahr oder mehr studiert haben , leichter Jobs .
Boyd (2018)	A 0 3||R:OTHER||Dahingegen||REQUIRED||-NONE-||0
	A 4 5||U:PNOUN||||||REQUIRED||-NONE-||0
	A 5 6||R:NOUN||BA-Studenten||REQUIRED||-NONE-||0
Ours	A 0 2||R:ADV ADV -> ADV||Dahingegen||REQUIRED||-NONE-||0
	A 2 3||U:PUNCT||||||REQUIRED||-NONE-||0
	A 5 5||M:PUNCT|||-||REQUIRED||-NONE-||0
	('On the other hand, BA students who have studied an extra year or more find jobs more easily again.')

Figure 4: Examples of grammatical error annotation for Czech and German

labels. We applied this approach to German (Boyd, 2018) and Czech (Náplava et al., 2022) to assess whether structured, interpretable annotations could still be produced in the absence of custom heuristics.

As shown in Figure 4, our system improves clarity by attaching POS information to word order and punctuation errors, allowing more consistent cross-lingual comparisons. For Czech, the example highlights differences in word order (WO) annotation: our method distinguishes between auxiliary and main verbs by incorporating POS information, whereas prior work generally treated such cases as generic WO errors. By capturing the syntactic function of the words involved, our method enables more precise and interpretable annotation. A similar improvement is seen for German, where our universal framework avoids language-specific categories while maintaining clear and consistent labeling.

Compared to previous implementations, our annotation outputs remain broadly consistent in terms of overall operation counts, with only minor variations, as shown in Table 2. This suggests that our universal framework based on the MRU scheme can replicate established annotation distributions. Table 3 lists the most frequent error annotations produced by our system alongside those from previous implementations. Our system makes the syntactic categories involved in each replacement edit explicit (R:P_1 → P_2), reflecting a different annotation choice rather than a direct re-labeling of existing tags.

Future work could explore how to map between these representations to support compatibility and facilitate comparative evaluations. Another direction is to extend this annotation scheme to additional languages: because our framework leverages universal POS tags and dependency labels from stanza, it can be readily applied to the ten other languages in the MultiGEC dataset (Masciolini

et al., 2025a) without additional customization.

	Missing	Replacement	Unnecessary	Total
Czech				
Náplava et al. (2022)	693	3707	515	4915
Ours	695	3672	530	4897
German				
Boyd (2018)	1341	4406	638	6385
Ours	1310	4348	612	6270

Table 2: Comparison of operation counts (Missing, Replacement, Unnecessary) on the development sets of Czech (first 1000 sentences; Náplava et al., 2022) and German (Boyd, 2018).

4.3 Refining language-specific annotations for Korean

Previous research on Korean grammatical error annotation has relied on extensive linguistic resources (Yoon et al., 2023). However, grammatical errors in Korean often manifest at the morpheme level, as observed in L2 writing from the National Institute of Korean Language (NIKL) corpus. In contrast, prior error annotation approaches primarily operate at the word level, which aligns with our methodology. To ensure consistency in annotation, previous work established two priority rules for assigning a single error type to each word because of the potential ambiguity in error classification, particularly when multiple error types could apply to the same token: (i) INSERTION > DELETION > others, and (ii) WS (word segmentation = WB) > WO > SPELL > SHORTEN (incorrect contraction of a word) > PUNCTUATION > OTHERS.

Building on these foundations, we implement language-specific error types based on Algorithm 1 and refine the WB (word boundary) category by introducing two subtypes: WB:M for missing spaces and WB:U for extraneous spaces. The former occurs when spaces are absent between words, causing multiple words to merge into a single unit, which can obscure meaning and hinder readability. The latter arises when superfluous spaces are inserted between or within words, disrupting the

Czech				German			
Náplava et al. (2022)		Ours		Boyd (2018)		Ours	
Annotation	Count	Annotation	Count	Annotation	Count	Annotation	Count
DIACR	989	NOUN → NOUN	760	PUNCT	942	DET → DET	832
OTHER	834	VERB → VERB	465	SPELL	816	NOUN → NOUN	814
PUNCT	487	PUNCT	396	DET:FORM	693	PUNCT	800
SPELL	457	ADJ → ADJ	299	OTHER	670	ADJ → ADJ	466
VERB	271	PRON	161	ORTH	529	VERB → VERB	305
WO	227	DET → DET	101	ADP	348	DET	241
NOUN:INFL	209	ADV → ADV	100	ADJ:FORM	277	ADP → ADP	170
PRON	187	NOUN → ADJ	98	PRON	273	PRON	170
MORPH	177	PUNCT → PUNCT	95	NOUN:FORM	260	PRON → PRON	144
ORTH:CASING	124	ADP → ADP	94	DET	242	AUX → AUX	143

Table 3: Comparison of the top 10 most frequent error annotations on the development sets of Czech (first 1000 sentences; Náplava et al., 2022) and German (Boyd, 2018).

natural flow of the text.

Additionally, we extend grammatical error annotation to functional morphemes, categorizing errors into (i) postposition errors (ADP), (ii) verbal ending errors (PART), and (iii) honorific suffix errors (HON). These errors are further classified into missing (M), unnecessary (U), and incorrect usage (R). Figure 5 illustrates corrections from two annotators: the noun phrase 음식이 *eumsig-i* ('food.NOM') is replaced with 음식을 *eumsig-eul* ('food.ACC'), annotated as R:NOUN -> NOUN:ADP, reflecting a case marker correction. Similarly, 막였습니다 *magyeossseubnida* is replaced with 먹었습니다 *meogeossseubnida* ('ate'), which constitutes a spelling error due to phonetic and orthographic similarity.[4]

```
S 비행기 1음식이 안 3막였습니다 .
A 1 2|||R:NOUN -> NOUN:ADP|||음식을|||REQUIRED|||-NONE-|||0
A 3 4|||R:Orthographic|||먹었습니다|||REQUIRED|||-NONE-|||0
A 3 4|||R:VERB -> VERB|||맞았습니다|||REQUIRED|||-NONE-|||1
```

Figure 5: Examples from the Korean M2 file: *I didn't eat the airplane food* (Annotator 0), and *The airplane food didn't agree with me* (Annotator 1)

4.4 Integrating deeper customization for Chinese

Chinese grammatical error annotation presents unique challenges due to the lack of explicit word boundaries (Qiu et al., 2025). Previous systems (Zhang et al., 2022; Gu et al., 2025) adopt segmentation schemes based on different linguistic assumptions: for instance, LTP[5] emphasizes compound words as cohesive lexical units, whereas stanza, trained on the Chinese GSD treebank[6], adopts a

finer-grained, morpheme-level segmentation strategy that tends to split compound expressions into smaller units.

These design choices reflect distinct philosophies rather than flaws. However, segmentation differences can affect downstream grammatical error annotation, including both the token spans and the syntactic interpretation of the correction. For example, whether a multi-character expression like 为什么 *wèishéme* ('why') is treated as one token or multiple (为 什么) influences how missing or replacement errors are classified.

To illustrate the flexibility of our framework, we adopt an LTP-style segmentation approach, which aligns more closely with native speaker intuitions about lexical units in Chinese. While the default stanza pipeline uses GSD-style morpheme-level segmentation, our framework allows researchers to substitute this with alternative schemes, such as LTP's compound-word-based segmentation. This optional customization demonstrates that language-specific preprocessing decisions, such as tokenization granularity, can be adapted within our framework to better support accurate and interpretable error annotation.

We achieve this integration by re-annotating the Chinese GSD treebank with LTP-informed word boundaries and retraining stanza on this revised corpus. This ensures compatibility with our preferred segmentation standard while preserving the benefits of stanza's POS tagging and parsing pipeline. As shown in Figure 6, the resulting annotations show more consistent edit spans and error categories, especially in contexts where compound expressions are frequent.

Ultimately, this customization demonstrates the modularity of our framework: rather than enforcing a one-size-fits-all solution, we allow researchers to tailor tokenization to fit linguistic expectations, making the system more robust and adaptable

---

[4]Annotator 1 annotates a replacement with 맞았습니다 *maj-assseubnida* ('agree'), altering the meaning of the sentence. This highlights a potential challenge in grammatical error annotation—distinguishing between true errors and alternative valid expressions that change sentence semantics.

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

[6]https://github.com/UniversalDependencies/UD_Chinese-GSD

Chinese GSD-based WB (Gu et al., 2025):
```
S ... 解释 为 ₁₀什么 这样 的 情况 ...
A ...
A 10 11|||R:PROPN -> PRON VERB AUX|||什么 出现 了|||REQUIRED|||-NONE-|||0
Correction: ... 解释 为 什么 出现 了 这样 的 情况 ...
```

Our LTP-based WB:
```
S ... 解释 ₇为 ₈什么 这样 的 情况 ...
A ...
A 7 8|||R:ADP -> ADV VERB|||为什么 出现|||REQUIRED|||-NONE-|||0
A 8 9|||R:PROPN -> AUX|||了|||REQUIRED|||-NONE-|||0
Correction: ... 解释 为什么 出现 了 这样 的 情况 ...
          ... jiěshì wèishéme chūxiàn le zhèyàng de qíngkuàng ...
          ('... explain why this kind of situation has occurred ...')
```

Figure 6: Fragments of grammatical error annotation examples in Chinese with different word boundaries. Incorrect GSD-based segmentation of 为什么 *wèishéme* ('why') leads to misleading annotation 什么 出现 了 *shénme chūxiàn le* ('what has occurred'), while LTP-based segmentation 为什么 出现 *wèishéme chūxiàn* ('why occurred') provides an accurate representation.

across languages and segmentation conventions.

5 Conclusion

This work advances grammatical error annotation and evaluation by introducing a standardized, modular framework for multilingual grammatical error typology. Building upon the foundations of `errant`, we designed a two-tiered system that separates language-agnostic annotation from structured language-specific extensions. This approach supports consistency across typologically diverse languages while allowing targeted customizations when needed.

We reimplemented `errant` using `stanza` to provide broader multilingual support, and demonstrated that our system produces accurate and interpretable annotations in English. We then demonstrated how our framework can be applied to other languages with varying levels of customization. For European languages, we showed that our POS- and dependency-based system can generate reliable annotations without requiring language-specific classification modules. For Korean, we applied minor refinements to capture morphologically salient features such as postpositions and spacing errors. Finally, for Chinese, we demonstrated how deeper customization–through the integration of language-specific tokenization and retraining of NLP components–can be incorporated into our framework to support fine-grained, linguistically coherent error annotation.

By balancing consistency and flexibility, our framework enables scalable, interpretable, and reusable grammatical error annotation across languages. This supports more consistent evaluation and clearer cross-linguistic comparison in multilingual GEC research.

Limitations

While our framework presents a unified and extensible approach to multilingual grammatical error annotation, the implementations described in this paper are primarily intended to demonstrate its adaptability across different languages and levels of customization. A detailed analysis of annotation improvements, including task-specific gains and downstream evaluation effects, is left to future work.

Although we rely on existing NLP tools such as `stanza` for tokenization and parsing, which offer broad multilingual coverage and consistent annotation schemes, these tools are not explicitly optimized for processing noisy or learner-generated text. This may introduce variability in some edge cases, particularly in languages with complex morphosyntax or ambiguous word segmentation.

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.

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.

Christopher Bryant, Zheng Yuan, Muhammad Reza Qorib, Hannan Cao, Hwee Tou Ng, and Ted Briscoe. 2023. Grammatical Error Correction: A Survey of the State of the Art. *Computational Linguistics*, 49(3):643–701.

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.

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.

Yang Gu, Zihao Huang, Min Zeng, Mengyang Qiu, and Jungyeul Park. 2025. Improving Automatic Grammatical Error Annotation for Chinese Through Linguistically-Informed Error Typology. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 2781–2798, Abu Dhabi, UAE. Association for Computational Linguistics.

Charles Hinson, Hen-Hsen Huang, and Hsin-Hsi Chen. 2020. Heterogeneous Recycle Generation for Chinese Grammatical Error Correction. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2191–2201, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Katerina Korre, Marita Chatzipanagiotou, and John Pavlopoulos. 2021. ELERRANT: Automatic grammatical error type classification for Greek. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2021)*, pages 708–717, Held Online. INCOMA Ltd.

Katerina Korre and John Pavlopoulos. 2020. ERRANT: Assessing and improving grammatical error type classification. In *Proceedings of the 4th Joint SIGHUM Workshop on Computational Linguistics for Cultural Heritage, Social Sciences, Humanities and Literature*, pages 85–89, Online. International Committee on Computational Linguistics.

Jenni Marjokorpi. 2023. The relationship between grammatical understanding and writing skills in finnish secondary l1 education. *Reading and Writing*, 36(10):2605–2625.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, and Robert Östling. 2025a. The MultiGEC-2025 shared task on multilingual grammatical error correction at NLP4CALL. In *Proceedings of the 14th Workshop on Natural Language Processing for Computer Assisted Language Learning*, pages 1–33, Tallinn, Estonia. University of Tartu Library.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, Robert Östling, Kais Allkivi, Špela Arhar Holdt, et al. 2025b. Towards better language representation in natural language processing: A multilingual dataset for text-level grammatical error correction. *International Journal of Learner Corpus Research*.

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, Yuanbin Wu, Christian Hadiwinoto, and Joel Tetreault. 2013. The CoNLL-2013 Shared Task on Grammatical Error Correction. In *Proceedings of the Seventeenth Conference on Computational Natural Language Learning: Shared Task*, pages 1–12, Sofia, Bulgaria. Association for Computational Linguistics.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A Universal Part-of-Speech Tagset. In *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC-2012)*, pages 2089–2096, Istanbul, Turkey. European Language Resources Association (ELRA).

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D Manning. 2020. Stanza: A Python Natural Language Processing Toolkit for Many Human Languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 101–108, Online. Association for Computational Linguistics.

Mengyang Qiu, Qingyu Gao, Linxuan Yang, Yang Gu, Tran Minh Nguyen, Zihao Huang, and Jungyeul Park. 2025. Chinese grammatical error correction: A survey. *arXiv*.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021. A Simple Recipe for Multilingual Grammatical Error Correction. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 702–707, Online. Association for Computational Linguistics.

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.

Catherine G Van Beuningen, Nivja H De Jong, and Folkert Kuiken. 2012. Evidence on the effectiveness of comprehensive error correction in second language writing. *Language Learning*, 62(1):1–41.

Junrui Wang, Mengyang Qiu, Yang Gu, Zihao Huang, and Jungyeul Park. 2025. Refined Evaluation for End-to-End Grammatical Error Correction Using an Alignment-Based Approach. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 774–785, Abu Dhabi, UAE. Association for Computational Linguistics.

Soyoung Yoon, Sungjoon Park, Gyuwan Kim, Junhee Cho, Kihyo Park, Gyu Tae Kim, Minjoon Seo, and Alice Oh. 2023. Towards standardizing Korean Grammatical Error Correction: Datasets and Annotation. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6713–6742, Toronto, Canada. Association for Computational Linguistics.

Min Zeng, Jiexin Kuang, Mengyang Qiu, Jayoung Song, and Jungyeul Park. 2024. Evaluating Prompting Strategies for Grammatical Error Correction Based on Language Proficiency. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 6426–6430, Torino, Italy. ELRA and ICCL.

Yue Zhang, Zhenghua Li, Zuyi Bao, Jiacheng Li, Bo Zhang, Chen Li, Fei Huang, and Min Zhang. 2022. MuCGEC: a Multi-Reference Multi-Source Evaluation Dataset for Chinese Grammatical Error Correction. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 3118–3130, Seattle, United States. 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.