

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

<http://open-writing-evaluation.github.io>

## Multilingual GEC Annotation Framework

- **Goal:** Develop a consistent, reusable framework for grammatical error annotation across typologically diverse languages.
- **Inspired by** the dataset-agnostic design of errant; extend its core philosophy to multilingual settings.
- **Two-component architecture:** a shared architecture that applies across languages (MRU: Missing, Replacement, and Unnecessary), and optional extensions tailored to language-specific features
- **Structured templates** for common error types (e.g., spelling, word order, word boundary) facilitate reuse across related languages.
- **Built on** the stanza NLP toolkit for tokenization and POS tagging in 70+ languages.
- **Easily extensible:** New languages can be supported with minimal adaptation.
- **Available** at [https://github.com/open-writing-evaluation/jp\\_errant\\_bea](https://github.com/open-writing-evaluation/jp_errant_bea).

## Examples of Error Annotation for Czech and German

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 labels.

Czech Náplava et al. (2022)	S Mám velkou rodinu , tak nemohla jsem mít naději , že něco dostanu .
Ours	A 5 7     R:WO     jsem nemohla     REQUIRED     -NONE-     0 (‘I have a big family, so I couldn’t hope to get anything.’)
German Boyd (2018)	S Dagegen wieder , bekommen BA Studenten die ein extra Jahr oder mehr studiert haben , leichter Jobs .
Ours	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 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.’)

## Language-Specific Error Annotation

```
1: function ERRORCLASSIFICATION (S, T):
2:   if (SIM (phonetic) > α1) ∧ (SIM (shape) > α2) then
3:     return R:SPELL:PHONOGRAPHIC
4:   else if (SIM (phonetic) > α1) then
5:     return R:SPELL:PHONETIC
6:   else if (SIM (shape) > α2) then
7:     return R:SPELL:SHAPE
8:   else if (SET (S) == SET (T)) then
9:     return R:WO
10:  else if (MERGE(S) == MERGE(T)) then
11:    return R:WB
12:  end if
13:  return {R}
```

The algorithm above presents our proposed classification routine for Replacement errors. Given a pair of word sequences—the source ( $S$ ) and the target ( $T$ )—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  $\alpha_1$  and  $\alpha_2$  govern sensitivity to phonetic and visual matches, respectively.

- The classification uses the following notation:
- $S, T$ : word sequences in the source and target sentences.
  - $SIM(phonetic)$  and  $SIM(shape)$ : similarity functions comparing pronunciation and visual form.
  - $SET(S)$ : returns a bag-of-words representation of  $S$ , disregarding word order.
  - $MERGE(S)$ : reconstructs a character sequence from the tokenized input (i.e., merging tokens without spaces) to test for boundary alignment.

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

R:SPELL:PHONETIC	<i>their</i> → <i>there</i>
R:WO	<i>You can help me</i> → <i>Can you help me</i>
R:WB	<i>ice cream</i> → <i>icecream</i>



Download the paper →

