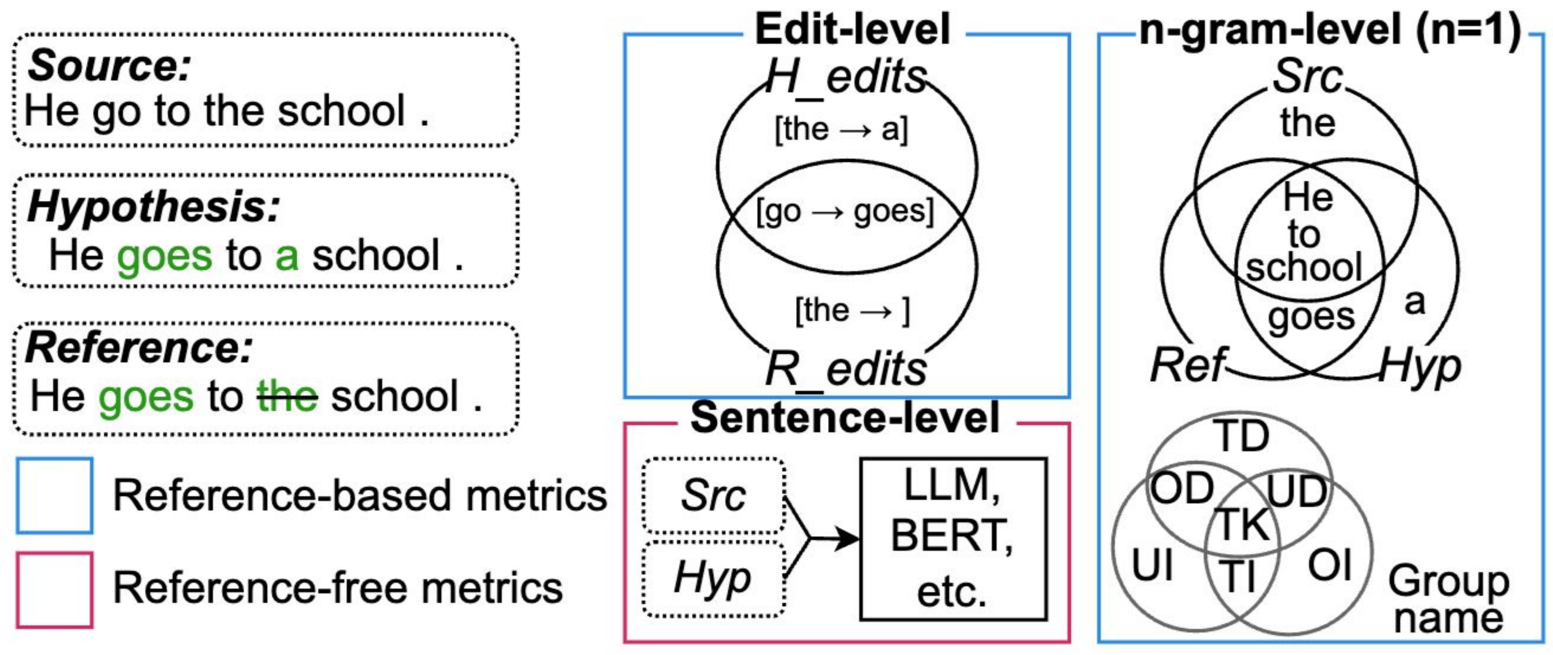
gec-metrics:
A Unified Library for Grammatical Error Correction Evaluation

Takumi Goto, Yusuke Sakai, Taro Watanabe *NARA Institute of Science and Technology*

https://github.com/gotutiyan/gec-metrics

Background

- Various GEC metrics have been proposed
  - e.g., {Edit, n-gram, sentence}-level
- **Users** want to use various metrics easily
- **Developers** want to develop new metrics and perform meta-evaluation easily



Problems of existing implementations

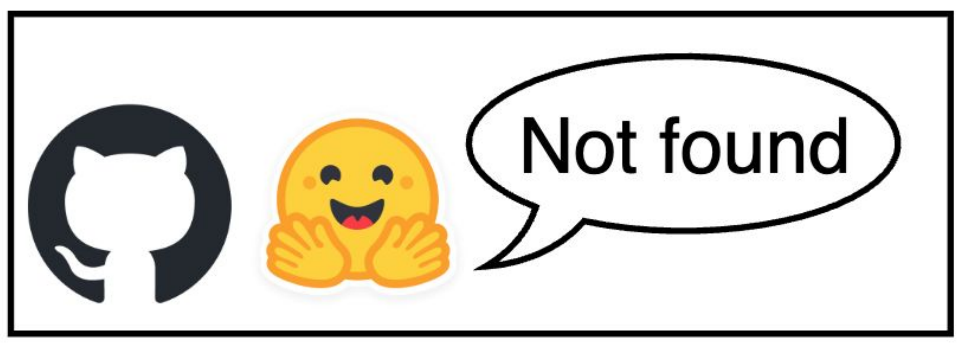
1. Inconsistent interfaces

The more metrics are used, the higher the experimental cost

```
python metric1.py --src <> --hyp <>
python metric2.py \
    --source <> --hypothesis <>
```

2. Lack of official resources

This limits reproducibility



3. Lack of API support

This limits future extensions and applications

```
from metric import Metric
metric = Metric()
s = metric.score(srcs=[...], hyps=[...])
```

gec-metrics: A unified library of various metrics

- gec-metrics supports various metrics within the **unified interface**, especially focusing on **API usage**

Features for users (line 4-14)

- Support both corpus-level and sentence-level scoring
  - Simple interface with string input and float output
  - All metrics have the same interface
- Nine metrics are supported currently
  - GLEU, GREEN, ERRANT, PT-ERRANT, GoToScorer, SOME, Scribendi, IMPARA, LLM metrics.

Features for developers (line 16-27)

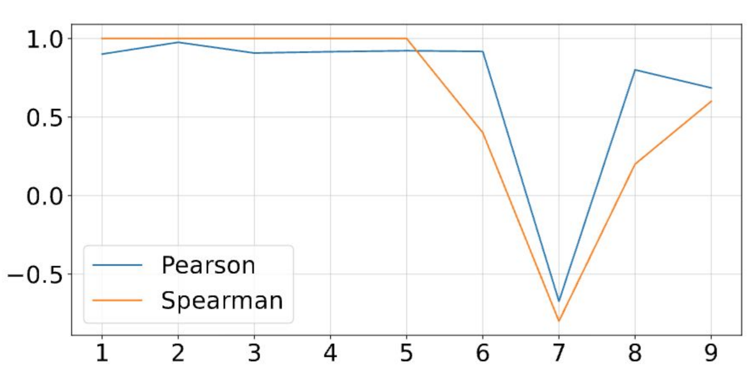
- Support both system- and sentence-level meta-evaluation
  - Receive a metric instance and output correlations
- Two meta-evaluation datasets are supported
  - GJG15: Human ranking for CoNLL-2014 submission systems
  - SEEDA: 14 systems includes recent neural models

```
1  from gec_metrics.metrics import ERRANT
2  from gec_metrics.meta_eval import
     MetaEvalSEEDA
3  metric = ERRANT(ERRANT.Config(beta=0.5))
4  SRCS = ["He go to the school."] * 100
5  HYPS = ["He goes to the school."] * 100
6  REFS = [["He goes to school."] * 100]
7
8  # Corpus-level scoring
9  system_score: float = metric.score_corpus(
10   sources=SRCS, hypotheses=HYPS,
       references=REFS
11 )  # Output: 0.833
12 # Sentence-level scoring
13 sent_score: list[float]  =
     metric.score_sentence(sources=SRCS,
     hypotheses=HYPS, references=REFS
14 )  # Output: [0.833, 0.833, ...]
15
16 ### Meta-evaluation on SEEDA ###
17 meta = MetaEvalSEEDA(
18   MetaEvalSEEDA.Config(system='base')
19 )
20 # System-level meta-evaluation
21 meta_system = meta.corr_system(metric)
22 print(f"SEEDA-S: {meta_system.ts_sent}")
23 # Output: MetaEvalBase.Corr(pearson=0.539,
     spearman=0.342)
24 # Sentence-level meta-evaluation
25 meta_sentence = meta.corr_sentence(metric)
26 print(f"SEEDA-S: {meta_sentence.sent}")
27 # Output:  MetaEvalBase.Corr(accuracy=0.594,
     kendall=0.188)
```

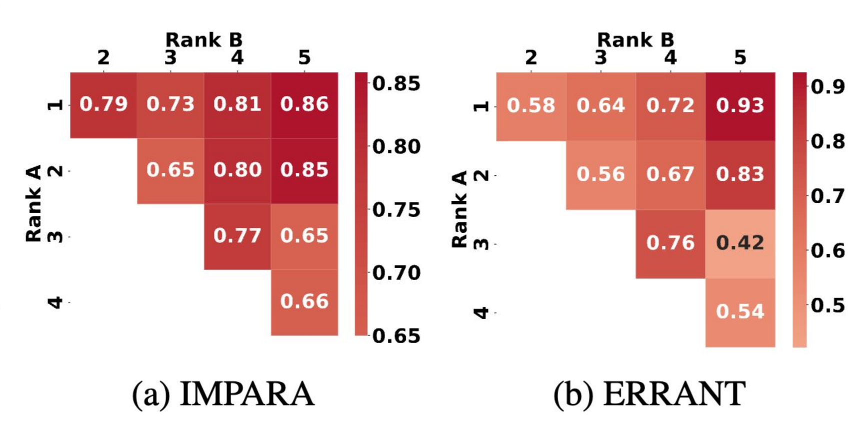
Experiments and Results

- Using above metrics and datasets, we conducted meta-evaluation
- Basically, we were able to reproduce the results reported in the original papers.
  - But the LLM metrics *[Kobayashi+ BEA2024]* were hard to be reproduced
- *gec-metrics* also provides analysis functions
  - **Window-analysis** *[Kobayashi+ TACL2024]* **:** Correlation in the stricted systems
  - **Pairwise Comparison:** Detailed results of sentence-level meta-evaluation by decomposing the results into rankings of human's rank pair.

	GJG15		SEEDA-S +Fluency setting	
Metrics	**Pearson**	**Spearman**	**Pearson**	**Spearman**
ERRANT	0.647	0.687	-0.592	-0.156
PT-ERRANT	0.704	0.786	-0.548	0.077
GLEU	0.706	0.626	0.155	0.543
GREEN	0.786	0.720	0.185	0.569
SOME	**0.957**	**0.923**	**0.931**	0.916
IMPARA	0.956	0.885	0.887	0.938
LLM-based metrics [Kobayashi+ BEA2024]				
GPT-4-E	0.383	0.357	-0.817	-0.393
GPT-4-S	-0.073	-0.181	0.322	0.613
Gemini-S	-0.205	-0.318	0.461	0.714
Qwen2.5-S	-0.247	-0.274	0.788	**0.942**



IMPARA results



(a) IMPARA (b) ERRANT