Automated Run-on Sentence Detection and Correction for Educational Writing

Chakraborty, S., Tian, Y., Banawan, M., Potter, A., Huynh, L., Yajjapurapu, Y., & McNamara, D. S.

Grammatical Error Correction
learning engineering
natural language processing
Writing Analytics

Run-on sentences, including fused sentences, comma splices, and conjunctive adverb misuse, pose a persistent challenge in student writing, undermining both human evaluation and automated analyses in learning environments. Despite their instructional importance, run-ons are underrepresented in major grammatical error correction (GEC) benchmarks. We present a two-stage NLP pipeline for run-on detection and minimal-change correction, designed within the Learning Engineering Framework to improve writing feedback while preserving student voice. Early annotation of 251 student sentences identified 29 potential run-ons, informing our pipeline design and human validation workflows.

Introduction

Run-on sentences represent a high-impact yet underexplored error type in educational writing. Within the Learning Engineering Framework(Goodell & Kolodner, 2023), this work creates data-informed systems that improve writing feedback while preserving student voice.

Run-on sentences occur in three primary subtypes: Comma Splice: Two independent clauses joined only by a comma without a coordinating conjunction (e.g., “I studied all night, I still failed the exam”). Fused Sentence: Two independent clauses with no punctuation or conjunction between them (e.g., “She finished her essay she submitted it online”). Conjunctive Adverb Misuse: A comma incorrectly placed before a conjunctive adverb connecting two independent clauses (e.g., “The results were promising, however more research is needed”).

Despite their instructional importance, run-ons are underrepresented in major GEC benchmarks such as BEA-2019 and CoNLL-2014 (Bryant et al., 2019), which emphasize broad error categories over clause-boundary errors common in student writing. Consequently, AI writing tools inadequately address run-ons, and no systematic framework exists for modeling this error type with minimal, voice-preserving correction.

Approach

Two-Stage Pipeline Architecture

We present a two-stage pipeline guided by iterative prototyping, structured data instrumentation, and linguistic theory.

Stage 1: Run-on Detection. The first stage focuses on classifying sentences as run-ons and identifying subtype categories. We employ transformer encoders (BERT, RoBERTa, DeBERTa) fine-tuned on annotated student writing samples. The detection model learns to recognize clause boundaries, punctuation patterns, and syntactic structures indicative of run-on errors. Subtype classification enables targeted feedback that helps students understand the specific nature of their error, moving beyond generic "grammar error" notifications toward pedagogically meaningful guidance.

Stage 2: Minimal-Change Correction. The second stage generates corrections using sequence-to-sequence models (T5, FLAN-T5, LLaMA-3) fine-tuned on paired error-correction examples. The key design principle is minimal-change correction: producing the smallest grammatical modification necessary to resolve boundary errors while preserving the rhetorical intent and stylistic choices of the student writer. Each run-on subtype allows multiple valid corrections, and the model selects the most contextually appropriate one while preserving minimal edit distance from the original text.. This approach respects student voice and avoids over-correction that could alter meaning or discourage developing writers.

Data Instrumentation

Training data derives from Canvas submissions, student essays, and discussion posts collected through Arizona State University’s Learning at Scale (L@S) research infrastructure. Initial annotation uses GPT-based few-shot prompting followed by human expert validation. Trained annotators review machine-generated annotations to confirm or reject run-on classifications, verify subtype labels, identify edge cases requiring adjudication, and evaluate correction quality for voice preservation. This human-in-the-loop workflow ensures annotation quality while enabling efficient processing of large text collections. Inter-annotator agreement is measured using Cohen's Kappa to ensure reliability, with disagreements resolved through expert adjudication to create gold-standard labels. Subtype balance is maintained through targeted sampling and data augmentation to address class imbalance inherent in naturalistic writing samples.

Early Insights and Preliminary Annotation

To support early design decisions for the detection and correction pipeline, we conducted an initial study using GPT-4-turbo with few-shot prompting to generate preliminary annotations for 251 student sentences.

Table 1.

Preliminary annotation results from initial study (n=251 sentences).

Category	Count
Total sentences annotated	251
Potential run-ons identified	29
Detection rate	11.6%

The model identified 29 potential run-on sentences distributed across the three subtypes. Example constructions identified include comma splices in argumentative essays where students connect related claims, fused sentences in narrative passages with rapid action sequences, and conjunctive adverb misuse with transitional words like "however" and "therefore." These annotations help surface challenging constructions, prototype subtype heuristics, and refine human-in-the-loop validation workflows. The relatively low detection rate (11.6%) aligns with expectations for student writing at the undergraduate level, though validated rates will emerge from ongoing human annotation efforts.

Implications for Learning Engineering

This project contributes to learning engineering through four areas: Data Instrumentation, by providing robust annotation protocols for an underrepresented clause-boundary error; Learner-Sensitive Design, by prioritizing minimal-change corrections that known preserve student voice; Educational NLP Infrastructure, by delivering validated resources and a modeling pipeline that support writing analytics in instructional contexts; and FERPA Compliance, by ensuring all processing occurs on secure university infrastructure without external API calls.

Conclusion and Next Steps

We have presented a two-stage pipeline for run-on sentence detection and correction in educational writing, designed within the Learning Engineering Framework. Our preliminary annotation study identified 29 potential run-ons in 251 student sentences, informing pipeline design and validation workflows. Next steps include: (1) completing human expert validation to establish ground truth labels, (2) fine-tuning transformer models on validated annotations, (3) evaluating detection precision and recall against human judgments, and (4) assessing correction quality through both automated metrics and human evaluation of voice preservation. This work addresses a significant gap in educational NLP by targeting an error type that is common in student writing yet underrepresented in existing GEC systems.

Acknowledgments

The research reported here was supported by the Institute of Education Sciences, U.S. Department of Education, through Grant R305N210041 and R305T240035 to Arizona State University. The opinions expressed are those of the authors and do not represent views of the Institute or the U.S. Department of Education.

References

Bryant, C., Felice, M., Andersen, Ø. E., & Briscoe, T. (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 (pp. 52–75). Association for Computational Linguistics. https://doi.org/10.18653/v1/W19-4406

Goodell, J., & Kolodner, J. (Eds.). (2023). Learning engineering toolkit: Evidence-based practices from the learning sciences, instructional design, and beyond. Taylor & Francis.

Omelianchuk, K., Atrasevych, V., Chernodub, A., & Skurzhanskyi, O. (2020). GECToR – Grammatical error correction: Tag, not rewrite. In Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications (pp. 163–170). Association for Computational Linguistics. https://doi.org/10.18653/v1/2020.bea-1.16