Adapting LLMs for Minimal-edit Grammatical Error Correction

Ryszard Staruch
Adam Mickiewicz University
Center for Artificial Intelligence
ryszard.staruch@amu.edu.pl

Filip Graliński
Adam Mickiewicz University
filipg@amu.edu.pl
Snowflake
filip.gralinski@snowflake.com

Daniel Dzienisiewicz
Adam Mickiewicz University
dzienis@amu.edu.pl

Abstract

Decoder-only large language models have shown superior performance in the fluency-edit English Grammatical Error Correction, but their adaptation for minimal-edit English GEC is still underexplored. To improve their effectiveness in the minimal-edit approach, we explore the error rate adaptation topic and propose a novel training schedule method. Our experiments set a new state-of-the-art result for a single-model system on the BEA-test set. We also detokenize the most common English GEC datasets to match the natural way of writing text. During the process, we find that there are errors in them. Our experiments analyze whether training on detokenized datasets impacts the results and measure the impact of the usage of the datasets with corrected erroneous examples. To facilitate reproducibility, we have released the source code used to train our models.[1]

1 Introduction

Grammatical Error Correction (GEC) is a Natural Language Processing task that covers the detection and correction of errors in texts. Current state-of-the-art models are either Sequence-to-Edit (Seq2Edit) models (encoder-only Transformers) that are trained to tag erroneous tokens and apply proper changes to them (Omelianchuk et al., 2020), or Sequence-to-Sequence (Seq2Seq) models (encoder-decoder Transformers) that are trained to generate the correct version of a given text (Rothe et al., 2021).

Over the years, two main directions have been established in GEC research: minimal-edit GEC and fluency-edit GEC (Bryant et al., 2023). The former focuses on applying only the minimal changes necessary to make the text grammatical and error-free. In contrast, fluency-edit GEC goes beyond minimal corrections to achieve native-language fluency.

[1] github.com/richardxoldman/llms-for-minimal-gec

Current decoder-only large language models (LLMs) achieve state-of-the-art performance on many NLP tasks. Instruction-tuned LLMs are able to produce high-quality texts and correct errors in the zero-shot approach, even without task-specific fine-tuning (Davis et al., 2024). On the JFLEG dataset (Napoles et al., 2017), which is a fluency-edit GEC dataset, the GPT3 and GPT4 models are capable of producing state-of-the-art results (Loem et al., 2023; Coyne et al., 2023). LLMs were also used by the winners of the recent multilingual grammatical error correction shared task – MultiGEC-2025 (Masciolini et al., 2025).

However, for a minimal-edit GEC, there is only one research work that reports better results compared to other solutions on English minimal-edit GEC benchmarks (Liang et al., 2025). The problem encountered by LLMs can be explained by the phenomenon of overcorrection (Fang et al., 2023).

To further explore LLMs adaptation for minimal-edit GEC, there is a need to find solutions that could allow LLMs to produce more strict outputs. Junczys-Dowmunt et al. (2018) by exploring the error rate adaptation topic show that neural network based solutions need more erroneous examples. Their experiments show that removing the correct examples leads to greater recall. Our intuition is that for modern LLMs, which are able to produce fluent corrections with high linguistic freedom even in the zero-shot manner, the opposite direction is needed, as there is a need for higher precision.

Sun and Wang (2022) propose a method for a precision-recall trade-off that requires beam-search decoding, which increases computational resources and inference time compared to greedy decoding. To overcome this issue, we propose a novel training schedule method to control the precision-recall trade-off during training instead of inference. Our method allows for the application of standard greedy decoding during inference without the need

Proceedings of the 20th Workshop on Innovative Use of NLP for Building Educational Applications, pages 118–128
July 31 - August 1, 2025 ©2025 Association for Computational Linguistics

...to a cafe and ~~and~~ I drank a drink.
I recommend you **to** practise any sport...
She is **one** of the ones that...
Sometimes we go to part~~y~~ies in the city.
...and I was very **happy** to hug him because I miss him...

Table 1: Examples of changes in target texts made during detokenization process by the Llama 3 70b model. Deletions are highlighted with a strikethrough, and insertions are highlighted in bold.

for external tools or algorithms to control the inference process.

Since LLMs are trained on raw texts and existing GEC datasets are available in word-tokenized (henceforth referred to as "tokenized") format (Bryant et al., 2023), it forces models to switch from working on raw texts to tokenized texts.

Another case that would require detokenized texts is any work that leverages probability distributions for language models, for example the Scribendi Score reference-less metric (Islam and Magnani, 2021).

To solve this issue, we detokenize the most common GEC datasets and verify whether training models on detokenized texts leads to better results. The detokenization process involved the usage of the LLM, during which we discovered that even the most popular datasets contain errors in annotations. We make the detokenized datasets available to the public to make them accessible to other researchers[2].

In summary, our contributions in this work are as follows:

- The LLM that achieves the state-of-the-art single-model system on the BEA-19 Shared Task test set.
- The study of error rate adaptation in the context of LLMs.
- The novel training schedule method that enables control of the precision-recall trade-off during training.
- The detokenization of the most common English GEC datasets, and the detailed analysis of annotation errors in them.

2 Datasets and their detokenization

The most common GEC datasets for English are available in a tokenized format due to evaluation tools that use the M2 format (Dahlmeier and Ng, 2012) such as ERRANT (Bryant et al., 2017). LLMs are trained on raw texts, so the tokenization process forces them to switch to the tokenized text and also to learn the tokenization process. To solve this issue, we detokenize FCE-train (Yannakoudakis et al., 2011), W&I+LOCNESS train and dev part (hereafter, we refer to the train split of this dataset as BEA-train, the dev split as BEA-dev and the test split as BEA-test) (Bryant et al., 2019) CoNLL-2014-test (Ng et al., 2014), and JFLEG datasets — these are the datasets we decided to use in our work, as they are one of the most commonly used GEC resources (Bryant et al., 2023). The statistics about them are given in the Appendix.

For the FCE-train, BEA-train, and BEA-dev datasets, the source texts were available in the raw format (the only work needed was to properly split them line by line). To detokenize the target texts of these datasets, we used the Sacremoses Detokenizer[3], but it did not correctly detokenize all the examples.

To improve the detokenization process, we leveraged the Llama-3.1-70b-Instruct model (denoted as Llama 3 70b), where the model task was only to detokenize the target text. We included a source text that is properly detokenized in the prompt to help the model in the detokenization process. The prompt is given in the Appendix.

In order to detokenize the CoNLL-2014 input texts, we had to properly split paragraphs at the sentence level, which are available in SGML format. We did this using a simple Python script with split rules and then manually adjusted examples that were not properly handled by the script.

For the JFLEG dataset we only had to detokenize inputs of the dataset, since the dataset has only dev and test splits. Due to the small size of the JFLEG dataset, we used Sacremoses Detokenizer and then manually adjusted the texts.

It should be emphasized that our work does not affect the examples in the test sets. The source texts for both the BEA-test and the CoNLL-2014-test were unchanged. The BEA-test target texts are hidden on the CodaLab platform and are not available publicly. There was no need to detokenize

[2] github.com/richardxoldman/detokenized-gec-datasets
[3] pypi.org/project/sacremoses/

Dataset	modified	essential	optional	erroneous	not assessable	**wrong annotations** (estimated lower bound)
BEA-dev	6.52%	80.77%	2.80%	12.59%	3.85%	5.22%
BEA-train	6.22%	78.67%	4.90%	9.80%	6.64%	4.89%
FCE-train	8.42%	71.68%	12.24%	12.24%	3.85%	6.04%

Table 2: Details for annotations to examples changed by the Llama 3 70b model.

the CoNLL-2014-test target texts, since the scoring script uses the M2 format to compute the results. It makes outcomes based on our version of the datasets fully comparable to the previous research.

The results reported on our version of the BEA-dev dataset may differ slightly from those reported by other researchers due to the changes described in Section 2.1, but are intended to select the most promising model, not to report the final results.

2.1 Incorrect annotations in datasets

In less than 10% of the examples, the Llama 3 70b model, when used for detokenization, occasionally modified the text beyond simply removing spaces in the correct version of the text. Table 1 shows examples of differences between the target texts in the dataset and the changes made by the Llama 3 70b model. Our initial investigation showed that those changes are mostly errors that were not corrected by a human annotator. Given this, we decided to do a manual annotation of such samples.

For our annotation purposes, the considered sentences were assigned four labels: *essential*, *optional*, *erroneous* and *not assessable*.

The *essential* label was assigned to sentences in which corrections were necessary and actually contributed to improving their accuracy.

The *optional* label was attributed to sentences in which the corrections made were not necessary, as their original versions were considered correct as well (e.g. sentences originally written in capital letters, which were then changed to lower case).

The *erroneous* label refers to situations where the corrections either do not fix the original mistakes in the sentences or create new mistakes in sentences that were already correct.

Finally, the *not assessable* label is used to mark corrections for which the quality, for various reasons, cannot be assessed by the annotator.

For BEA-dev, all examples (284) modified by the Llama 3 70b model were verified, whereas for the other two datasets, random samples of the same size (284 examples) were checked. The results of the annotation process are shown in Table 2.

2.2 Detokenization impact

To verify whether the detokenization process and the modification of examples by the Llama 3 70b model have an impact on the GEC models, we decided to train the LLMs on the FCE-train and the BEA-train datasets in four different processing setups:

1. **detokenized-filtered**: Detokenized datasets **excluding** examples modified by the Llama 3 70b model.

2. **tokenized-filtered**: Tokenized datasets corresponding to the examples that remained unmodified in the detokenized version.

3. **detokenized-full**: Detokenized datasets **including** all examples, both modified and unmodified.

4. **tokenized-full**: Tokenized datasets corresponding to the full set of detokenized examples (original, untouched datasets).

Please note that **tokenized-*** setups refer to the original examples ”as is”, without any modifications introduced by the Llama 3 70b model.

The **detokenized-filtered** setup compared to the **tokenized-filtered** setup shows whether the detokenization process has an impact on the models' performance, since both models are fine-tuned on the same examples with the same hyperparameter setup. The details about the hyperparameters are given in the Appendix.

The ***-full** setups against the ***-filtered** setups show whether the changes made by the Llama 3 70b model in the datasets have an impact on the results, because the **detokenized-full** setup contains the modified examples by the Llama 3 70b model,

Model	Size	Setup	BEA-dev			JFLEG-dev
			P	R	$F_{0.5}$	GLEU
Qwen 2.5	1.5B	detokenized-filtered	57.90	42.10	53.86	56.10
Qwen 2.5	1.5B	tokenized-filtered	59.00	38.48	53.31	56.17
Qwen 2.5	1.5B	detokenized-full	57.86	42.75	54.04	56.22
Qwen 2.5	1.5B	tokenized-full	59.92	37.79	53.63	56.01
Llama 3 Small	3B	detokenized-filtered	63.34	47.52	59.39	57.42
Llama 3 Small	3B	tokenized-filtered	63.31	47.29	59.29	57.58
Llama 3 Small	3B	detokenized-full	63.04	48.32	59.42	57.56
Llama 3 Small	3B	tokenized-full	62.61	46.22	58.46	56.96
Gemma 2	9B	detokenized-filtered	68.84	56.40	65.93	58.70
Gemma 2	9B	tokenized-filtered	68.84	55.90	65.79	58.99
Gemma 2	9B	detokenized-full	69.07	57.13	66.30	58.72
Gemma 2	9B	tokenized-full	69.86	55.67	66.47	58.40

Table 3: Results for different dataset processing setups.

Dataset	M	R	U
BEA-dev	50.74%	38.87%	10.39%
BEA-train	46.93%	40.28%	12.79%
FCE-train	61.33%	31.96%	6.71%

Table 4: Details about the operations performed by the Llama 3 70b model. The labels stand for: Missing, Replacement and Unnecessary.

whereas the **tokenized-full** setup contains all the original examples (also the erroneous ones). Again, the number of training examples is the same, but the difference lies in the quality of the annotations in examples that were changed by the Llama 3 70b model.

All models were trained for one epoch on the FCE-train dataset and then for one epoch on the BEA-train dataset. In this and subsequent experiments, we report the results for the BEA-dev and JFLEG-dev datasets, since these datasets give a view for both minimal-edit and fluency-edit GEC. Table 3 presents the results for 3 different LLMs of different sizes: Qwen2.5-1.5B-Instruct (denoted as Qwen 2.5), Llama-3.2-3B-Instruct (denoted as Llama 3 Small) and gemma-2-9b-it (denoted as Gemma 2).

2.3 Results analysis

The results show that LLMs can learn the tokenized version of the texts and in some cases even achieve better metric scores compared to the models trained on the detokenized texts. We can see that there are no clear gains in terms of $F_{0.5}$ score from using the detokenized version of datasets.

The transition from the **tokenized-filtered** to the **tokenized-full** setup increases precision in each experiment but lowers recall and GLEU values. In all cases, transition from the **detokenized-filtered** setup to the **detokenized-full** setup improves recall and slightly improves the GLEU score. It shows that the changes made by the Llama 3 70b model result in outputs with higher linguistic freedom, which is expected, since the most common change made by the Llama 3 70b model is the Missing operation (Table 4), while using the original sentences makes the models produce more strict outputs.

We can also see that the size of the models significantly impacts the results. Therefore, for the next experiments we will further explore the Gemma 2 model, as it is the best performing model. Although Gemma 2 achieves the best $F_{0.5}$ score on the **tokenized-full** setup, the next experiments will be performed on the detokenized version of the datasets, as they contain corrected erroneous annotations. The other reason is that our systems can be used in the work of other researchers who need a model that produces detokenized output. It would be also simply practical in terms of using the system in the environment where the output does not require removing the unnecessary spaces.

3 Overcorrection problem

In the minimal-edit GEC task, the goal is to find and correct only those parts of the texts that are clearly erroneous, without making further improvements to their fluency. Due to the pre-training goal

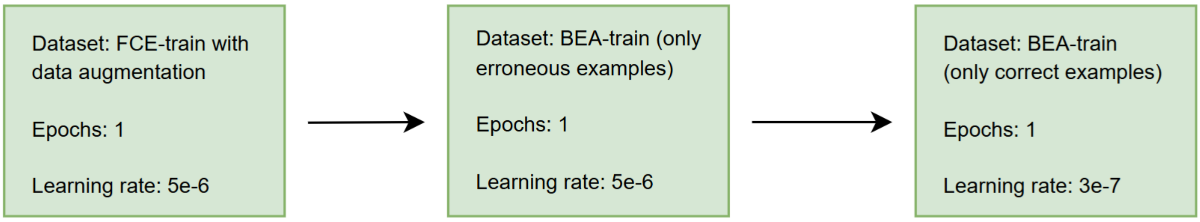

Figure 1: Visualization of the fine-tuning process for our best performing Gemma 2 model on the BEA-dev dataset.

of LLMs, which is to maximize the probability of the next token, and the flexibility they gain from instruction fine-tuning process, LLMs tend to produce more fluent output. While this characteristic may be advantageous for fluency-edit GEC, the objective of minimal-edit GEC is to apply only the minimal necessary corrections.

Standard minimal-edit GEC benchmarks, which are based on texts written by English language learners, put a greater weight on precision than on recall, because suggesting an incorrect change is considered more negative than ignoring an error (Ng et al., 2014). Therefore, a proper adaptation of the model is needed to correct errors with high precision.

For the Chinese minimal-edit GEC, Yang and Quan (2024) proposes an alignment model which is used to filter only minimal corrections from the initial correction, which may be fluent.

One of the most recent works proposes the novel method for LLM fine-tuning, Edit-Wise Preference Optimization (EPO) that fits the minimal-edit GEC task better than the standard supervised fine-tuning (SFT) approach (Liang et al., 2025). In our work, we explore the SFT approach with a focus on the datasets rather than the different training approaches, and show that proper data preprocessing or training schedule can lead to the successful minimal-edit LLM model.

4 Data augmentation

During GEC model fine-tuning, datasets play a crucial role in the whole process. One of the most important attributes of the GEC datasets is the error rate. The common practice for neural models that are trained from scratch is to remove unedited pairs (Chollampatt and Ng, 2018; Kiyono et al., 2019), because for these models there is a need for improved recall.

Large language models produce fluent output with high recall, which may suggest that removing unedited pairs for LLMs is unnecessary and could worsen the results. Furthermore, it may be possible that providing additional unedited pairs could improve minimal-edit error correction for LLMs.

To provide more real examples that may not be fluent, but are still acceptable, we propose a data augmentation method to split each example (consisting of source text and corrected text) into two pairs. The new pair is created by using the corrected text as both the source and target text. For example, the sentence pair “Alice have a cat.” and “Alice has a cat.” can be split into the following examples:

- Alice have a cat. $\rightarrow$ Alice has a cat.
- Alice has a cat. $\rightarrow$ Alice has a cat.

Our method can be applied to any dataset and does not require any additional models/tools to extend a given dataset.

5 Training schedule

Current approaches to GEC training scheduling consist of dividing data into 2 or 3 groups based on data quality and then training a model in the correct order, from the lowest quality data to the highest (Bout et al., 2023). We follow this approach, but to control the precision-recall trade-off, we propose to extend it even further.

In the final stage (with the highest quality dataset – in our case it is BEA-train dataset), we split the data into two groups. The first group contains only erroneous texts, whereas the second group contains only correct examples. During the stage, we first train the model on the first group (only erroneous examples), and then we train the model on the second group (only correct examples) with lower learning rate. Figure 1 shows the step-by-step training schedule for the best performing model on the BEA-dev set.

Our intuition behind this approach is that a model first learns how to correct errors and later is tuned to understand which examples are correct but

Dataset processing approach	Erroneous sentences		BEA-dev			JFLEG-dev
	FCE-train	BEA-train	P	R	$F_{0.5}$	GLEU
ONLY-ERRONEOUS	100%	100%	60.74	58.79	60.34	58.16
UNCHANGED	65.43%	69.02%	68.99	57.12	66.24	58.73
AUGMENTED	39.55%	40.83%	71.42	53.42	**66.92**	58.21

Table 5: Results for the dev sets for the experiment with our data augmentation method.

in some cases not perfectly fluent. During the last stage, when the model is fine-tuned only on correct examples, the model only learns to not apply corrections to texts.

Choosing a proper learning rate value (or number of examples) enables controlling the precision-recall trade-off in LLMs, as lowering learning rate should make the model learn not to correct more smoothly while still being able to correct the errors in texts.

6 Experiments

6.1 Data augmentation experiments

To test whether the addition of unedited pairs can positively affect LLMs in the minimal-edit GEC task, we train the Gemma 2 model[4] with the same hyperparameter setup as in the experiment from Section 2 in three different dataset processing approaches:

- only erroneous examples (denoted as **ONLY-ERRONEOUS**)
- erroneous examples + unedited examples (denoted as **UNCHANGED**)
- erroneous examples + unedited examples + unedited examples created from erroneous examples by applying our data augmentation method (denoted as **AUGMENTED**)

As in the previous experiment, we first train one epoch on the FCE-train dataset and then one epoch on the BEA-train dataset.

Table 5 shows the results on the BEA-dev and JFLEG-dev datasets. We can see that unedited examples are needed to improve the LLMs performance. Even on the fluency-edit dataset, the scores are better when unedited pairs are added to the dataset (the **UNCHANGED** approach). For the **AUGMENTED** approach, the $F_{0.5}$ score is the highest among all approaches, but the GLEU score is lower compared to the **UNCHANGED** approach.

This study shows that lowering the error rate in the GEC datasets is a way to make LLMs produce minimal-edit outputs. It also shows that when new solutions are available, such as modern LLMs, approaches or practices from previous research, such as removing unedited pairs, should be reevaluated and tested again.

6.2 Training schedule experiment

We also carried out an experiment with different learning rate values for the last group (only correct examples) for our training schedule method for the Gemma 2 model. We also test whether applying our data augmentation method for the FCE-train dataset improves the results.

Note that in this experiment data augmentation method is **not** applied to the BEA-train dataset.

Table 6 shows how the precision-recall trade-off depends on the learning rate value. It can be observed that even small changes in the learning rate value noticeably influence the trade-off, making the hyperparameter very sensitive.

When applying the data augmentation method for the FCE-train dataset, the BEA-dev set $F_{0.5}$ score can be improved compared to the best value achieved in the previous experiment (the **AUGMENTED** dataset processing approach).

Although the data augmentation method was designed to enhance precision, we observe that results with data augmentation on the FCE-train have higher recall. In this experiment, we hypothesize that training on the FCE-train provides general GEC knowledge, while fine-tuning on the BEA-train determines the model's behavior in terms of the precision-recall trade-off as model is first fine-tuned on erroneous examples and then on the correct ones.

[4]For the data augmentation and training schedule experiments we also tested the gemma-2-9b-it-SimPO model and achieved slightly better results, but we decided to use the original Gemma 2 model as our goal is not to maximize the benchmark scores.

Learning rate	FCE-train Augmented	BEA-dev			JFLEG-dev
		P	R	$F_{0.5}$	GLEU
1e-7	✗	65.90	58.18	64.19	58.58
1e-7	✓	65.10	58.33	63.62	58.60
2e-7	✗	69.30	56.05	66.17	58.64
2e-7	✓	69.22	56.40	66.21	58.66
2.5e-7	✗	70.94	53.73	66.67	58.47
2.5e-7	✓	70.96	54.40	66.89	58.28
3e-7	✗	73.63	48.72	66.80	57.60
3e-7	✓	73.52	50.10	**67.23**	57.90
3.5e-7	✗	75.81	44.92	66.65	56.74
3.5e-7	✓	75.38	46.82	67.18	57.35
4e-7	✗	77.49	40.15	65.34	55.48
4e-7	✓	76.74	43.49	66.57	56.15
5e-7	✗	79.74	24.79	55.29	50.26
5e-7	✓	78.88	31.78	60.85	52.91

Table 6: Results for the dev sets for the experiment with our training schedule method.

Figure 1 shows the complete training process for the model with the highest $F_{0.5}$ score.

6.3 Results on the test datasets

From each experiment, we choose the most promising model based on its performance on the BEA-dev dataset to evaluate it on the BEA-test, CoNLL-2014-test, and JFLEG-test datasets. In Table 7, Gemma 2 Augmentation refers to the best model from Section 6.1 (only applying the data augmentation method) and Gemma 2 Training-Schedule refers to the best model from Section 6.2.

Table 7 shows that our model from the training-schedule experiment achieves a new state-of-the-art single model result on the BEA-test dataset and has competitive results with other solutions on the CoNLL-2014-test dataset. It should be noted that our models were trained only on two relatively small datasets, whereas other solutions were trained on a much larger number of examples, except for the Mistal-7b-EPO model.

To get more insights about the impact of the different model selection on the results, we also performed a single experiment with the gemma-2-27b-it and llama-2-13b-chat (Gemma 2 (27b) Training-Schedule and LLama-2-13b Training-Schedule in the tables) models with the same training schedule and hyperparameters as the best performing model on the BEA-dev dataset, so the model training is exactly the same as for the Gemma 2 Training-Schedule model.

The Llama-2-13b achieves even worse results than these reported by (Omelianchuk et al., 2024). It can be explained by using different datasets during fine-tuning process. The precision and recall are both worse than those of the Gemma 2 model. This suggests that model size is not the only important factor; other details about the LLM, such as its novelty, architecture, and the dataset used for training, also matter.

The Gemma 2 (27b) achieves even a better score than the best Gemma 2 9b model on the BEA-test set, but it may be slightly overtuned for precision due to the same learning rate value in the final stage with the bigger model, which can be observed in the worse results for the CoNLL-2014-test dataset.

Table 8 shows the results for the JFLEG-test dataset. We can see that even if our models are fine-tuned for minimal-edit GEC, they achieve a higher score than the average of the scores computed for the JFLEG-test references. It suggests that LLMs can find a proper balance between minimal-edit GEC and fluency-edit GEC.

7 Conclusions

Our work demonstrates that there are several ways to fine-tune an LLM for minimal-edit grammatical error correction, without the need for pre-training them on a large number of examples. We propose easy-to-implement methods for controlling the precision-recall trade-off during fine-tuning.

Moreover, we show that choosing a more recent

Model	Size	CoNLL-2014-test			BEA-test		
		P	R	$F_{0.5}$	P	R	$F_{0.5}$
T5 Large (Rothe et al., 2021)	700M	-	-	66.04	-	-	72.06
T5 XL (Rothe et al., 2021)	3B	-	-	67.65	-	-	73.92
T5 XXL (Rothe et al., 2021)	11B	-	-	68.75	-	-	75.88
GECToR (Tarnavskyi et al., 2022)	355M	74.40	41.05	64.00	80.70	53.39	73.21
TemplateGEC (Li et al., 2023)	770M	74.80	50.00	68.10	76.80	64.80	74.10
FLAN-T5 XXL (Cao et al., 2023)	11B	75.00	53.80	69.60	78.80	68.50	76.50
DeCoGLM (Li and Wang, 2024)	335M	75.10	49.40	68.00	77.40	64.60	74.40
BART Base (Wang et al., 2024)	400M	76.20	52.20	69.80	77.70	67.50	75.40
Llama-2-13b (Omelianchuk et al., 2024)	13B	77.30	45.60	67.90	74.60	67.80	73.10
Mistral-7b-EPO (Liang et al., 2025)	7B	76.71	52.56	**70.26**	78.16	68.07	75.91
Gemma 2 Augmentation	9B	73.80	**56.16**	69.43	74.86	**71.35**	74.13
Gemma 2 Training-Schedule	9B	75.74	51.47	69.24	79.87	68.90	77.41
Llama-2-13b Training-Schedule	13B	71.07	50.11	65.59	74.10	67.54	72.69
Gemma 2 (27b) Training-Schedule	27B	**77.38**	47.88	68.89	**82.28**	67.03	**78.70**

Table 7: Single model results for the minimal-edit GEC test sets.

Model	GLEU
Source (Uncorrected)	40.54
Reference (Average)	62.37
Conv Seq2Seq (Ge et al., 2018)	62.42
Transformer (Stahlberg and Kumar, 2021)	64.70
GPT-3.5 (Coyne et al., 2023)	63.40
GPT-4 (Coyne et al., 2023)	**65.02**
Gemma 2 Augmentation	63.72
Gemma 2 Training-Schedule	62.91
Llama-2-13b Training-Schedule	62.53
Gemma 2 (27b) Training-Schedule	62.42

Table 8: Results for the fluency-edit GEC dataset (JFLEG-test).

LLM is also an important factor that impacts the overall performance of the model. The Gemma 2 9b model, even as a smaller model achieved much better performance compared to the Llama-2-13b model.

The detokenization process did not improve model performance, but our findings on the errors in the most common GEC datasets show the need for a proper curation of datasets. Our work also shows that LLMs can be effectively used as a detokenization tool.

8 Limitations

Our work covers only experiments on English GEC datasets, so it would be beneficial to extend the research to check how LLMs would perform in other languages. We did not conduct experiments on other types of models. It is hard to tell whether our methods would improve the Seq2Seq or Seq2Edit approaches.

The other issue is that we applied only greedy decoding during inference. The results could be even better if different decoding methods were applied. It would also be worth comparing these methods applied in LLMs with the Seq2Seq or Seq2Edit models.

The reusability of the training schedule method is limited by the requirement for extensive learning rate tuning for any different model or dataset due to high sensitivity to minor changes in learning rate.

Obtaining the highest $F_{0.5}$ might be considered overfitting for a specific test set and evaluation metric, but in practical terms, the style of grammar correction depends on specific needs, guidelines, etc., so this might be a desired behavior.

Lastly, running our models requires a lot of memory and computational power, so for many people it would be impossible to run them on their devices. Our models may not be practical for everyday use, but they can be used to create synthetic datasets that can be used to train smaller models.

References

Andrey Bout, Alexander Podolskiy, Sergey Nikolenko, and Irina Piontkovskaya. 2023. Efficient grammatical error correction via multi-task training and optimized training schedule. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 5800–5816, Singapore. 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*, pages 643–701.

Hannan Cao, Liping Yuan, Yuchen Zhang, and Hwee Tou Ng. 2023. Unsupervised grammatical error correction rivaling supervised methods. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 3072–3088, Singapore. Association for Computational Linguistics.

Shamil Chollampatt and Hwee Tou Ng. 2018. A multilayer convolutional encoder-decoder neural network for grammatical error correction. In *Proceedings of the Thirty-Second AAAI Conference on Artificial Intelligence and Thirtieth Innovative Applications of Artificial Intelligence Conference and Eighth AAAI Symposium on Educational Advances in Artificial Intelligence*, AAAI'18/IAAI'18/EAAI'18. AAAI Press.

Steven Coyne, Keisuke Sakaguchi, Diana Galvan-Sosa, Michael Zock, and Kentaro Inui. 2023. Analyzing the performance of gpt-3.5 and gpt-4 in grammatical error correction. *Preprint*, arXiv:2303.14342.

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.

Christopher Davis, Andrew Caines, O Andersen, Shiva Taslimipoor, Helen Yannakoudakis, Zheng Yuan, Christopher Bryant, Marek Rei, and Paula Buttery. 2024. Prompting open-source and commercial language models for grammatical error correction of English learner text. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 11952–11967, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Tao Fang, Shu Yang, Kaixin Lan, Derek F. Wong, Jinpeng Hu, Lidia S. Chao, and Yue Zhang. 2023. Is chatgpt a highly fluent grammatical error correction system? a comprehensive evaluation. *Preprint*, arXiv:2304.01746.

Tao Ge, Furu Wei, and Ming Zhou. 2018. Reaching human-level performance in automatic grammatical error correction: An empirical study. *Preprint*, arXiv:1807.01270.

Md Asadul Islam and Enrico Magnani. 2021. Is this the end of the gold standard? a straightforward reference-less grammatical error correction metric. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3009–3015, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Marcin Junczys-Dowmunt, Roman Grundkiewicz, Shubha Guha, and Kenneth Heafield. 2018. Approaching neural grammatical error correction as a low-resource machine translation task. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 595–606, New Orleans, Louisiana. Association for Computational Linguistics.

Shun Kiyono, Jun Suzuki, Masato Mita, Tomoya Mizumoto, and Kentaro Inui. 2019. An empirical study of incorporating pseudo data into grammatical error correction. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1236–1242, Hong Kong, China. Association for Computational Linguistics.

Wei Li and Houfeng Wang. 2024. Detection-correction structure via general language model for grammatical error correction. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1748–1763, Bangkok, Thailand. Association for Computational Linguistics.

Yinghao Li, Xuebo Liu, Shuo Wang, Peiyuan Gong, Derek F. Wong, Yang Gao, Heyan Huang, and Min Zhang. 2023. TemplateGEC: Improving grammatical error correction with detection template. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 6878–6892, Toronto, Canada. Association for Computational Linguistics.

Jiehao Liang, Haihui Yang, Shiping Gao, and Xiaojun Quan. 2025. Edit-wise preference optimization for grammatical error correction. In *Proceedings of the 31st International Conference on Computational Linguistics*, pages 3401–3414, Abu Dhabi, UAE. Association for Computational Linguistics.

Mengsay Loem, Masahiro Kaneko, Sho Takase, and Naoaki Okazaki. 2023. Exploring effectiveness of

GPT-3 in grammatical error correction: A study on performance and controllability in prompt-based methods. In *Proceedings of the 18th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2023)*, pages 205–219, Toronto, Canada. Association for Computational Linguistics.

Arianna Masciolini, Andrew Caines, Orphée De Clercq, Joni Kruijsbergen, Murathan Kurfalı, Ricardo Muñoz Sánchez, Elena Volodina, and Robert Östling. 2025. 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.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. JFLEG: A fluency corpus and benchmark for grammatical error correction. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 229–234, Valencia, Spain. 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.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. GECToR – grammatical error correction: Tag, not rewrite. In *Proceedings of the Fifteenth Workshop on Innovative Use of NLP for Building Educational Applications*, pages 163–170, Seattle, WA, USA → Online. Association for Computational Linguistics.

Kostiantyn Omelianchuk, Andrii Liubonko, Oleksandr Skurzhanskyi, Artem Chernodub, Oleksandr Korniienko, and Igor Samokhin. 2024. Pillars of grammatical error correction: Comprehensive inspection of contemporary approaches in the era of large language models. In *Proceedings of the 19th Workshop on Innovative Use of NLP for Building Educational Applications (BEA 2024)*, pages 17–33, Mexico City, Mexico. Association for Computational Linguistics.

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.

Felix Stahlberg and Shankar Kumar. 2021. Synthetic data generation for grammatical error correction with tagged corruption models. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 37–47, Online. Association for Computational Linguistics.

Xin Sun and Houfeng Wang. 2022. Adjusting the precision-recall trade-off with align-and-predict decoding for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 686–693, Dublin, Ireland. Association for Computational Linguistics.

Maksym Tarnavskyi, Artem Chernodub, and Kostiantyn Omelianchuk. 2022. Ensembling and knowledge distilling of large sequence taggers for grammatical error correction. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3842–3852, Dublin, Ireland. Association for Computational Linguistics.

Yixuan Wang, Baoxin Wang, Yijun Liu, Qingfu Zhu, Dayong Wu, and Wanxiang Che. 2024. Improving grammatical error correction via contextual data augmentation. In *Findings of the Association for Computational Linguistics ACL 2024*, pages 10898–10910, Bangkok, Thailand and virtual meeting. Association for Computational Linguistics.

Haihui Yang and Xiaojun Quan. 2024. Alirector: Alignment-enhanced Chinese grammatical error corrector. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 2531–2546, Bangkok, Thailand. Association for Computational Linguistics.

Helen Yannakoudakis, Ted Briscoe, and Ben Medlock. 2011. A new dataset and method for automatically grading ESOL texts. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 180–189, Portland, Oregon, USA. Association for Computational Linguistics.

A Training details

We trained our <=13b models on a 2xA100 (80GB) GPU setup and the 27b model on a 4xA100 (80GB) GPU setup. We used 4xA100 (80GB) GPU setup to run the Llama 3 70b model for the detokenization process. A single model training took 2-3 hours. The hyperparameter values are described in Table 10. The following prompt was used during training our models and during inference:

Correct the following text, making only minimal changes where necessary.

Text to correct:

<source text>

Corrected text:

<target text>

B Detokenization prompt

The following prompt was used to detokenize the datasets:

You will receive two texts: source text and corrected text. Corrected text may not have proper spaces. Your task is to remove/add proper spaces to the corrected text. Do not write any comments, just write corrected text with proper spaces.

Source text: <source text>

Corrected text: <target text>

Only change spaces, you must not change punctuation.

Hyperparameter name	Value
learning rate	5e-6
batch size	4
gradient accumulation steps	4
warmup steps (for each dataset)	100
lr scheduler	linear
epochs (for each dataset)	1
optimizer	AdamW8bit
weight decay	0.01

Table 10: Hyperparameter values used to train our models.

Dataset	#Examples	Erroneous sentences
FCE-Train	28.4k	65.43%
BEA-train	34.3k	69.02%
BEA-test	4.5k	–
BEA-dev	4.4k	67.36%
CoNLL-2014-test	1.3k	71.90%
JFLEG-dev	754	95.36%
JFLEG-test	747	95.31%

Table 9: Details of the datasets used in our work. Note that there ratio of erroneous sentences could be different compared to the statistics about the datasets from different research works due to the changes made by the Llama 3 70b model during the detokenization process.