CS4248 Final Report

A0223021E, A0246220U, A0222480N, A0245173H, A0246099R
Group 47
Mentored by Naaman Tan
{e0564062, e0913139, e0560012, e0890735, e0909645}@u.nus.edu

Abstract

LLMs have achieved remarkable success in various natural language tasks. However, in the domain of grammar error correction, they often exhibit a tendency to overcorrect for fluency at the cost of authorial voice. In this paper, we address this challenge using Direct Preference Optimization. We construct and release two ranked preference datasets: one comprising grammar correction candidates generated by LLMs, and the other utilizing backtranslations ranked by edit distance. Through experimental evaluation, we demonstrate that LLM generated datasets offer a cost-effective strategy for preference alignment, reducing the need for costly human feedback. [1] [2]

1 Introduction

Grammar Error Correction (GEC) is the task of detecting and correcting grammatical errors in text. With the growing number of English speakers and learners worldwide, English Grammar Error Correction has attracted much attention as it has great applications in assisting learners in writing tasks. In recent years, GEC has improved significantly due to emerging Large Language Models (LLMs) like Generative Pre-trained Transformer (GPT) models. Despite their high performance, LLMs often generate more natural and contextually appropriate corrected text due to their contextual understanding of language, leading to redundant edits and a tendency towards over-correction (Fang et al., 2023). In some cases, we may prefer to achieve grammatical accuracy while still preserving authorial voice over improving fluency and coherence of our writing.

To address this gap, we aim to explore and analyze methods to train a Grammar Error Correction tool which prioritizes generating grammatically accurate text with minimal edits. Given a grammatically erroneous sentence, we can obtain a grammatically accurate sentence with minimal edits required on the input sentence.

Our contribution can be summarised as follows:

1. We train and release capable grammar corrector models that can correct spelling, punctuation, subject-verb agreement and other common grammatical errors, in some cases producing corrections that surpass the human annotated references.

2. We show that decoder-only transformers perform as well as encoder-decoder models when finetuned for grammar correction.

3. We curate and release a ranked dataset of candidate corrections for grammatically incorrect texts generated using backtranslation and ranked by edit distance from the original text for preference optimization.

4. We explore additional methods of preference alignment using rejection sampling and leveraging feedback from powerful LLMs like GPT-4 and Mixtral-8x7B.

2 Related Work

Transfer learning: Recent works have demonstrated the effectiveness of using a pre-trained mT5 model across a wide range of NLP tasks (Rothe et al., 2021a). Therefore, we leverage the generalization capabilities of pre-trained models to achieve high performance with comparatively less data and computational resources.

Neural Machine Translation: Neural machine translation (NMT) a natural extension of deep learning

[1] Source code can be found at: https://github.com/AY2324S2-CS4248-Team-47/Grammar-Error-Correction

[2] Our models and datasets can be found at: https://huggingface.co/AY2324S2-CS4248-Team-47

techniques, replaced the traditional Statistical Machine Translation (SMT) approach by employing a single, comprehensive neural network to handle the entire correction process. This eliminated the need for intricate feature engineering specific to GEC. Unlike SMT, NMT training is an end-to-end process, streamlining the system and removing the necessity for separately trained components. Notably, NMT has demonstrated state-of-the-art performance across various GEC tasks. (Flachs et al., 2021; Rothe et al., 2021b) It operates on the encoder-decoder framework (Cho et al., 2014), where the encoder initially processes and encodes the input sequence into hidden state representations, and the decoder subsequently generates the corrected output sequence by predicting each word based on the input sequence and previously generated words. Several network architectures, including RNNs, CNNs, and Transformers, have been proposed for constructing the encoders and decoders within NMT models (Bahdanau et al., 2014; Gehring et al., 2016; Vaswani et al., 2017).

Edit-based approaches: The edit generation approach in GEC focuses on producing a sequence of edits instead of correcting entire sentences, resulting in faster inference speeds compared to whole sentence correction methods (Stahlberg and Kumar, 2020). However, it primarily relies on token-based edits, which may struggle with complex fluency improvements (Lai et al., 2022). Edit generation can be approached either as a sequence tagging (Omelianchuk et al., 2020) or a sequence-to-sequence task (Stahlberg and Kumar, 2020). In the sequence tagging method, an edit operation is predicted for each token, while the sequence-to-sequence approach generates edits comprising a position span, replacement string, and optional edit tag, enhancing interpretability and performance. Despite slower inference due to left-to-right dependency, the sequence-to-sequence method remains significantly faster than whole sentence correction methods (Stahlberg and Kumar, 2020).

Direct Preference Optimization: Direct Preference Optimization (DPO) is a novel approach to training large language models. Unlike traditional training methods like maximum likelihood estimation (MLE), DPO trains the model to directly predict preferences between pairs of inputs rather than predicting the next token in a sequence. This is achieved by presenting the model with pairs of inputs and training it to determine which input is preferred over the other. By doing so, the language model effectively learns to optimize for desired outcomes directly, making it akin to a reward model. DPO aims to improve the language model's ability to generate outputs that are preferred by humans. It represents a significant advancement in language modeling, offering potential applications in various tasks where generating preferred outputs is essential. (Rafailov et al., 2024)

3 Evaluation Metrics

We mainly examined two factors to evaluate the performance of our models; the corrected text should preserve the meaning of the erroneous text, and the corrected text are obtained with minimal edits to the erroneous text. In order to compare and evaluate the performance across models, we used GLEU, ROUGE-L, BertScore and Levenshtein distance.

GLEU: General Language Understanding Evaluation (GLEU) scorer (Napoles et al., 2015, 2016) rewards correct edits (True Positives) by matching n-grams between the generated text and the human-annotated reference and penalizes for n-grams that overlap with the source text but not the reference (False Negatives). We compute average GLEU score of the model's corrections on the test data to evaluate the model's accuracy at correcting grammatical errors.

ROUGE: Recall-Oriented Understudy for Gisting Evaluation (ROUGE) scorer (Lin, 2004) measures the similarity between the generated text and the source text by measuring the recall of n-grams in the generated text. It is useful to evaluate how much of the important information in the source text is present the corrected text, ensuring that our GEC model sufficiently retains the meaning of the source text.

BertScore: BertScore (Zhang et al., 2019) measures semantic similarity between hypothesis and the reference based on contextual embeddings generated by a pre-trained BERT model . We computed the average BertScore of each model using the erroneous text as the reference and the model output as hypothesis to evaluate whether the model preserves meaning of text.

Levenshtein distance: Levenshtein distance (Levenshtein et al., 1966) quantifies the minimum number of single-character edits (insertions, deletions, or substitutions) required to transform one string into another. Since we aim to train a GEC model which makes minimal edits to the source text, we compute the Levenshtein distance between two sequences normalised by the length of the source text as a measure of edit distance.

To compare our models' performance we compute the average normalised Levenshtein distance of the model's corrections on the test data.

4 Methodology

We regard GEC task as a machine translation task with the grammatically erroneous text as the source language and grammatically accurate sentences as the target language. Given the success of Neural Machine Translation (NMT) for GEC tasks (Yuan and Briscoe, 2016), we adopted an NMT approach for our base model with exploration of a few commonly-used architecture for the encoder and decoder component, namely a Long Short-Term Memory (LSTM) recurrent neural network (RNN), Gated Recurrent Units (GRU) recurrent neural network and Transformer based models (Vaswani et al., 2017).

4.1 Training a Model from Scratch

Several preliminary approaches have been explored to address the issue of GEC. We evaluate various models as our baselines, trained on WI-LOCNESS dataset. We implemented the following GEC models as possible baseline models:

LSTM-attention and GRU-attention models: We trained simple two-layer LSTM and GRU models with an attention mechanism for 40 epochs.

Transformer model: We trained a T5 Small model from scratch for 100 epochs.

Despite the training, vanilla RNN models such as GRU and LSTM, as well as a T5 transformer exhibited sub-optimal performance, producing incoherent and repeating tokens.

4.2 Finetuning

For finetuning, we consider 2 pretrained decoder-only models - Phi-2B (Gunasekar et al., 2023) and StableLM-3B (Bellagente et al., 2024) which are state-of-the-art models with $\leq 3B$ parameters.

We formulate the task as a causal language modelling (next word prediction) task, and pass both the input and output concatenated together as the input to the model. Note that the decoder is trained using a triangular attention mask, so it cannot "cheat" by attending to future tokens $w_i, w_{i+1}...$ while predicting the i-th token.
We use the Stanford Alpaca format (Taori et al., 2023) for our model inputs with instruction "Rewrite the given text and correct grammar, spelling, and punctuation errors", grammatically incorrect text as input and corrected text as the response. Additionally, we use **loss masking** to backpropagate only the losses on the response (ie the corrections) produced by the model; ie we mask the losses for predictions within the grammatically incorrect input. We find that this improves learning considerably, and hypothesize that this is because without loss masking, the model needs to learn to predict grammatically incorrect text, and this part of the loss term tends to dominate the loss on the corrections.
Since all three models have billions of parameters, we quantize the base model and use LoRA (Hu et al., 2021) with rank stabilization (Kalajdzievski, 2023) for parameter efficient finetuning. This significantly reduces the compute required for training, reduces the chance of overfitting on our relatively small dataset and avoids catastrophic forgetting since the weights of the base model are unchanged.

4.3 Preference Dataset Construction

We explore two different methods to build a ranked preference dataset to encode the preference for minimal edits while correcting grammar.

4.3.1 Approach 1: Backtranslation

Drawing inspiration from backtranslation-based approaches for data augmentation in low-resource neural machine translation (Fadaee et al., 2017), we employ backtranslation to generate candidates for preference optimization.
Candidate generation: We employ a two-step process, utilizing a total of four languages: Chinese, French, Spanish and German. In each case, the text was first translated from English to one of these languages (step 1) and then translated back to English (step 2) using Google Translate API. This approach generates a diverse array of correction candidates, while ensuring that the candidates remain similar to the original text.
Ranking: We score these candidates by edit distance to the original (grammatically incorrect) text, and use a tournament-style ranking procedure to create a binary preference dataset, similar to methods discussed in (Zhao et al., 2023). Given a list of candidate m corrections, we rank pairs of them using edit distance, then rank the winners of these pairs and so on, repeating this procedure until $m - 1$ binary Accepted/Rejected pairs

are yielded.

4.3.2 Approach 2: LLM feedback

Our second approach is similar to Intel, 2024: we use large language models to generate correction candidates for training examples and rank based on model preference.

Candidate generation: We prompt GPT-4 (Achiam et al., 2023) and Mixtral-8x7B Instruct (Jiang et al., 2024) to correct training examples with as few corrections as possible using the same prompt template: *"Rewrite the given text without grammatical, spelling and punctuation errors. Make as few corrections as possible. Give only the corrected version of the text."*

Ranking: Since GPT-4 outperforms Mixtral in language understanding tasks, we assume that GPT-4's corrections are generally more reliable or closer to the desired grammatical standards. Therefore we binarize the dataset by accepting GPT-4's response and rejecting Mixtral's response.

We use Direct Preference Optimization (Rafailov et al., 2024) to directly optimize the preference model from the constructed pairs by maximizing the log-likelihood of predicting the accepted response and minimizing the log-likelihood of predicting the rejected response. We release both datasets on HuggingFace Hub for further research.

5 Experiments

5.1 Setup

Dataset: We trained on the Write & Improve dataset (Bryant et al., 2019) hosted on HuggingFace, which contains 3000 grammatically incorrect letters, stories, articles and essays in response to various prompts and their corresponding annotations. The dataset has two splits - train (3000 examples) and validation (300 examples). We further performed a train-test split on the train dataset with `test_size` 0.1 and `seed` 0. We truncated all input texts to a maximum of 512 tokens, choosing this limit based on the 95th percentile of input text size in the train dataset, which we found to be 491.

Baselines LSTM and GRU: We trained an RNN-based encoder-decoder architecture with two-layer LSTM and GRU and dot attention with 1024 hidden units, a dropout ratio of 0.2, and a gradient clipping value & a teaching force probability of 0.5. The models were trained using a single RTX 3090 GPU for 40 epochs, and training took more than 7 hours.[3]

Baseline transformer: We trained a T5 Small model ($60.5M$ parameters) using the HuggingFace Trainer. The model was trained using the AdamW optimizer (Loshchilov and Hutter, 2017) with 16-bit precision (`fp16`). A maximum learning rate of $5e-5$ was used with a linear learning rate scheduler. Training was done on a single RTX 3090 GPU for a maximum of 100 epochs and took between 1 - 1.5 hours.[4]

Finetuning: All models were quantized to 8 bits and trained using QLoRA with rank 64 (Dettmers et al., 2023). We use the `trl` library(von Werra et al.) in conjunction with Flash Attention (Dao, 2023) for training. The models were trained using the AdamW optimizer (Loshchilov and Hutter, 2017) in half-precision (`bfloat16` or `fp16`) format on a single RTX 3090 GPU for a maximum of 3 epochs, and training took between 1-2 hours.[5]

DPO: Similar to the finetuning step, we performed DPO on the finetuned model using LoRA (rank 64). The models were trained in `bfloat16` using the AdamW optimizer with 5% warmup steps, a maximum learning rate of $5e-5$ and a cosine annealing scheduler for a maximum of 2 epochs. $\beta = 0.1$ was used to control the model's divergence from the finetuned reference model. Training was done on a single RTX 3090 and took 3 hours.[6].

[3] https://github.com/AY2324S2-CS4248-Team-47/Grammar-Error-Correction/blob/main/gec_baseline_RNN.ipynb

[4] https://github.com/AY2324S2-CS4248-Team-47/Grammar-Error-Correction/blob/main/gec_baseline_transformer-train.ipynb

[5] https://github.com/AY2324S2-CS4248-Team-47/Grammar-Error-Correction/blob/main/sft_config.json

[6] https://github.com/AY2324S2-CS4248-Team-47/Grammar-Error-Correction/blob/main/dpo_config.json

6 Results and Discussion

Table 1: BertScore Metrics Comparison Across Models

Model	BertScore		
	Precision	Recall	F1
GRU-attention Baseline	0.9026	0.8724	0.8867
LSTM-attention Baseline	0.7927	0.8598	0.8237
Transformer Baseline	0.8144	0.8188	0.8165
Phi2 Baseline	0.9671	0.9595	0.9632
StableLM Baseline	0.9378	0.9536	0.9451
StableLM DPO LLM	0.9438	0.9571	0.9500
StableLM DPO Backtranslation	0.9372	0.9547	0.9453

Table 2: GLEU Comparison Across Models

Model	GLEU
GRU-attention Baseline	0.0971
LSTM-attention Baseline	0.1876
Transformer Baseline	0.0434
Phi2 Baseline	0.5874
StableLM Baseline	0.5713
StableLM DPO LLM	0.5728
StableLM DPO Backtranslation	0.5424

Table 3: ROUGE Metrics Comparison Across Models

Model	ROUGE-L
GRU-attention Baseline	0.4904
LSTM-attention Baseline	0.4200
Transformer Baseline	0.2193
Phi2 Baseline	0.8931
StableLM Baseline	0.8647
StableLM DPO LLM	0.8754
StableLM DPO Backtranslation	0.8602

Table 4: Average Edit Distance Metrics Comparison Across Models

Model	Edit distance
GRU-attention Baseline	-
LSTM-attention Baseline	0.796
Transformer Baseline	0.739
Phi2 Baseline	0.209
StableLM Baseline	0.101
StableLM DPO LLM	0.101
StableLM DPO Backtranslation	0.169

Baselines: Our LSTM and GRU models consistently outperform the baseline Transformer model significantly as shown in Tables 1, 2, 3. This may be because transformers generally need a lot more data for training when compared to LSTM and GRU architectures.

Finetuned vs baseline: As we would expect, models that have been previously pretrained on large corpora of data demonstrate superior performance compared to the baseline models which were trained from scratch.

Phi-2 vs StableLM: We find that the finetuned Phi-2 model consistently outperforms StableLM across all metrics. This is suprising to us, given that Phi-2 has been pretrained only on synthetic, textbook-like data. We expected that even though Phi-2 performs better in benchmarks, it would be less suited for our task since the prompts are grammatically incorrect, which would make it difficult for the model to understand as it has not been exposed to erroneous text in training.

DPO: We encountered CUDA related issues while running DPO training on Phi-2, so we chose to evaluate our DPO datasets on StableLM instead. We find that DPO using LLM generated candidates outperforms the finetuned model, while DPO using the backtranslated dataset performs worse. Our results demonstrate that using LLM feedback pairs for DPO training can improve performance with no additional overhead required for training a reward model from human feedback.

CEFR-level Performance

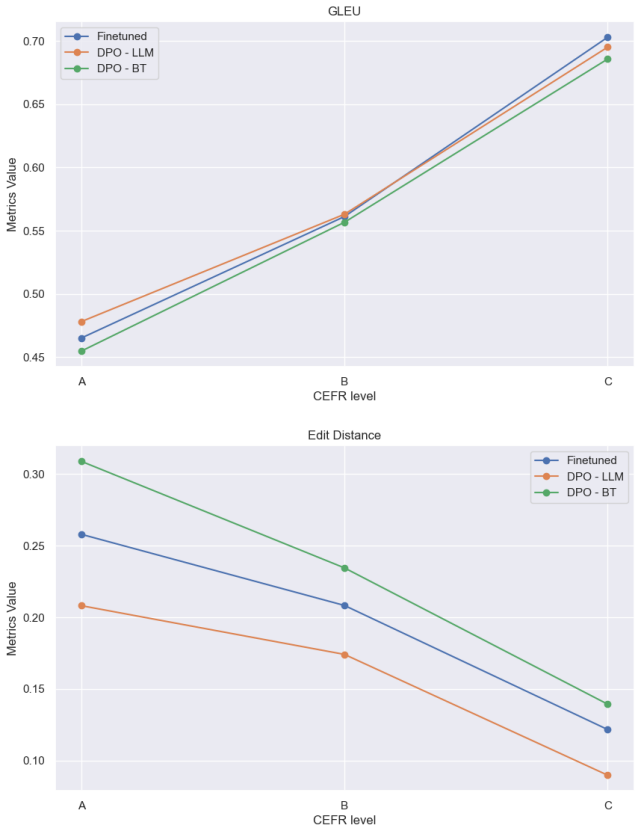

Figure 2: Edit distance by CEFR category

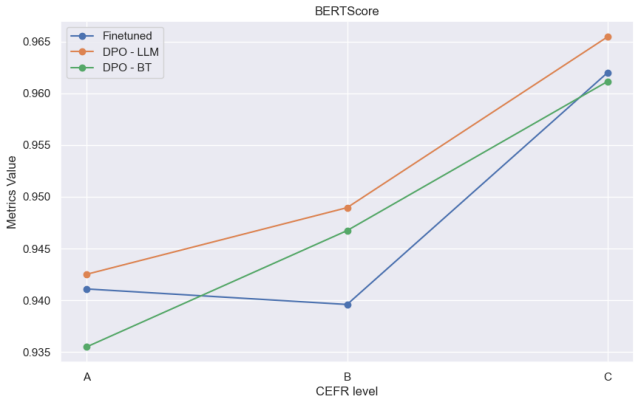

Figure 3: BERTScore by CEFR category

We analyse the performance of the StableLM models' grammar correction performance on different English proficiency levels as classified by Common European Framework of Reference for Languages (CEFR) labels provided in the W&I dataset. The various CEFR levels are as follows:

Level A: Basic or beginner proficiency
Level B: Independent or intermediate proficiency
Level C: Advanced proficiency
We observe that the performance of all models improve with an increase in the CEFR levels. We hypothesize this is because in texts of higher English proficiency, there errors would be easier to rectify and lower in number. So, the model would understand the original text better, to perform corrections.

7 Conclusion

In conclusion, we show that it is possible to mitigate the problem of over-corrections in LLMs and preserve authorial voice using preference aligment methods such as DPO. Additionally, we show that LLM-generated datasets can offer a cost-effective alternative to human-annotated datasets. However, our models still struggle with in lower proficiency texts where grammatical errors are more frequent and varied, pointing to a gap in training regimes that fail to adequately represent less formal language uses. Furthermore, training on synthetic datasets generated using LLMs risks reinforcing or amplifying biases in downstream applications. Sampling from a variety of LLMs with different prompts and careful validation of the generated data are imperative to mitigate this issue.

8 Statement of independent work

1. Declaration of Original Work. By entering our Student IDs below, we certify that we completed our assignment independently of all others (except where sanctioned during in-class sessions), obey ing the class policy outlined in the introductory lecture. In particular, we are allowed to discuss the problems and solutions in this assignment, but have waited at least 30 minutes by doing other activities unrelated to class before attempting to complete or modify our answers as per the class policy. We have documented our use of AI tools (if applicable) in a following table, as suggested in the NUS AI Tools policy. This particular document did not use any AI Tools to proofcheck and was constructed and edited purely by manual work.

A0223021E, A0246220U, A0222480N, A0245173H, A0246099R
e0564062, e0913139, e0560012, e0890735, e0909645

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Marco Bellagente, Jonathan Tow, Dakota Mahan, Duy Phung, Maksym Zhuravinskyi, Reshinth Adithyan, James Baicoianu, Ben Brooks, Nathan Cooper, Ashish Datta, et al. 2024. Stable lm 2 1.6 b technical report. *arXiv preprint arXiv:2402.17834*.

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.

Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Tri Dao. 2023. Flashattention-2: Faster attention with better parallelism and work partitioning. *arXiv preprint arXiv:2307.08691*.

Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. 2023. Qlora: efficient finetuning of quantized llms (2023). *arXiv preprint arXiv:2305.14314*.

Marzieh Fadaee, Arianna Bisazza, and Christof Monz. 2017. Data augmentation for low-resource neural machine translation. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 567–573, Vancouver, Canada. 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. *arXiv preprint arXiv:2304.01746*.

Simon Flachs, Felix Stahlberg, and Shankar Kumar. 2021. Data strategies for low-resource grammatical error correction. In *Proceedings of the 16th Workshop on Innovative Use of NLP for Building Educational Applications*, pages 117–122.

Jonas Gehring, Michael Auli, David Grangier, and Yann N Dauphin. 2016. A convolutional encoder model for neural machine translation. *arXiv preprint arXiv:1611.02344*.

Suriya Gunasekar, Yi Zhang, Jyoti Aneja, Caio César Teodoro Mendes, Allie Del Giorno, Sivakanth Gopi, Mojan Javaheripi, Piero Kauffmann, Gustavo de Rosa, Olli Saarikivi, et al. 2023. Textbooks are all you need. *arXiv preprint arXiv:2306.11644*.

Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*.

Intel. 2024. Orca dpo pairs. https://huggingface.co/datasets/Intel/orca_dpo_pairs.

Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. 2024. Mixtral of experts. *arXiv preprint arXiv:2401.04088*.

Damjan Kalajdzievski. 2023. A rank stabilization scaling factor for fine-tuning with lora. *arXiv preprint arXiv:2312.03732*.

Shaopeng Lai, Qingyu Zhou, Jiali Zeng, Zhongli Li, Chao Li, Yunbo Cao, and Jinsong Su. 2022. Type-driven multi-turn corrections for grammatical error correction. *arXiv preprint arXiv:2203.09136*.

Vladimir I Levenshtein et al. 1966. Binary codes capable of correcting deletions, insertions, and reversals. In *Soviet physics doklady*, volume 10, pages 707–710. Soviet Union.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81.

Ilya Loshchilov and Frank Hutter. 2017. Decoupled weight decay regularization. *arXiv preprint arXiv:1711.05101*.

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.

Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault. 2016. Gleu without tuning. *arXiv preprint arXiv:1605.02592*.

Kostiantyn Omelianchuk, Vitaliy Atrasevych, Artem Chernodub, and Oleksandr Skurzhanskyi. 2020. Gector–grammatical error correction: tag, not rewrite. *arXiv preprint arXiv:2005.12592*.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021a. 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.

Sascha Rothe, Jonathan Mallinson, Eric Malmi, Sebastian Krause, and Aliaksei Severyn. 2021b. A simple recipe for multilingual grammatical error correction. *arXiv preprint arXiv:2106.03830*.

Felix Stahlberg and Shankar Kumar. 2020. Seq2edits: Sequence transduction using span-level edit operations. *arXiv preprint arXiv:2009.11136*.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. 2023. Alpaca: A strong, replicable instruction-following model. *Stanford Center for Research on Foundation Models. https://crfm. stanford. edu/2023/03/13/alpaca. html*, 3(6):7.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, and Nathan Lambert. TRL: Transformer Reinforcement Learning.

Zheng Yuan and Ted Briscoe. 2016. Grammatical error correction using neural machine translation. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–386.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2019. Bertscore: Evaluating text generation with bert. *arXiv preprint arXiv:1904.09675*.

Yao Zhao, Rishabh Joshi, Tianqi Liu, Misha Khalman, Mohammad Saleh, and Peter J Liu. 2023. Slic-hf: Sequence likelihood calibration with human feedback. *arXiv preprint arXiv:2305.10425*.