Proofread: Fixes All Errors with One Tap

Renjie Liu*, **Yanxiang Zhang***, **Yun Zhu***, **Haicheng Sun, Yuanbo Zhang, Michael Xuelin Huang, Shanqing Cai, Lei Meng, Shumin Zhai**
Google Inc.

Abstract

The impressive capabilities in Large Language Models (LLMs) provide a powerful approach to reimagine users' typing experience. This paper demonstrates the Proofread feature in Gboard, a virtual keyboard running on mobile phones. Proofread enables seamless sentence-level and paragraph-level corrections with a single tap. We describe the complete system in this paper, from data generation, metrics design to model tuning and deployment. To obtain models with sufficient quality, we implement a careful data synthetic pipeline tailored to online use cases, design multifaceted metrics, employ a two-stage tuning approach to acquire the dedicated LLM for the feature: the Supervised Fine Tuning (SFT) for foundational quality, followed by the Reinforcement Learning (RL) tuning approach for targeted refinement. Specifically, we find sequential tuning on Rewrite and proofread tasks yields the best quality in SFT stage, and propose global and direct rewards in the RL tuning stage to seek further improvement. Extensive experiments on a human-labeled golden set showed our tuned PaLM2-XS model achieved 85.56% good ratio. We launched the feature to Pixel 8 devices by serving the model on TPU v5 in Google Cloud, with thousands of daily active users. Serving latency was significantly reduced by quantization, bucket inference, text segmentation, and speculative decoding. Our demo could be seen in Youtube.

1 Introduction

Gboard is an statistical-decoding-based keyboard on mobile devices developed by Google. Decoding (Ouyang et al., 2017) is necessary due to the error-prone process of "fat finger" touch input on small screens. According to Azenkot and Zhai (2012), the per-letter error rate is around 8%-9% without decoding.

Gboard provides various error correction features, some active (automatic) and other passive (require the user's further manual action and selection) to provide a smooth typing experience (Ouyang et al., 2017). Active key correction (KC), and active auto correction (AC), word completions and next-word predictions support the users to type the current word and next word by fixing typos and providing multiple word candidates in the suggestion bar or inline (smart compose). Post correction (PC) supports fixing errors in last one or more committed words. Furthermore, The more passive Spell Checker and Grammar Checker supported by small-sized logistic regression and seq2seq models respectively detect the possible errors in committed sentences and mark them with red underlines, users can fix the errors by clicking the incorrect words and commit the correct words from the displayed candidates.

There are two types of user experience limitations with the existing correction approaches. First, users still have to type relatively slowly and accurately to avoid making too many or too severe errors that the small (but instantly fast) on-device correction models such as KC, AC and PC cannot handle due to their limited ability to model longer-span context. Second, users need to manually engage in the multi-step passive correction features, such as the grammar checker and the spell checker, to correct the committed words one after another.

Supervising the committed words while typing and fixing errors sequentially by editing after commit take users' cognitive and visual-motor resources and slow down their typing speed. One desired pattern of fast typing users of Gboard is to focus on keyboard only without checking the committed words while typing. To this end, a high quality sentence-or-higher-level correction feature is often called for, in order to help those "fast and sloppy" users who prefer to focus on typing then

*Equal contribution, alphabetical order. Correspondence to {renjieliu,zhangyx,yunzhu}@google.com.

Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 286–293

switch to error corrections at a higher level.

In this paper, we propose the **Proofread** feature to alleviate the pain points of fast typers by providing the sentence-level and paragraph-level error fixes with only one-tap. Proofread falls into the area of Grammatical Error Correction (GEC), which has a long history of research from rule-based to statistical approaches to neural network models (Bryant et al., 2023). The astonishing capability growth of Large Language Models (LLMs), offers a new opportunity to unlock the high quality sentence-level grammar fixes.

We present the entire system to tune and serve the LLM model behind Proofread in this paper. The system consist of four parts, data generation, metrics design, model tuning and model serving. Firstly, dataset is generated by a carefully designed error synthetic framework which integrates errors frequently made on keyboard to simulate the users' input, several further steps are conducted to ensure the data distribution is close to Gboard domain maximally. Secondly, several metrics are designed to measure the model from various dimensions. As the answers are always not unique specifically for long examples, the metric combined with grammar error existence check and same meaning check based on LLMs are considered as the key metrics for comparing the model quality. Thirdly, inspired by InstructGPT (Ouyang et al., 2022), Supervised Fine-tuning followed by the Reinforcement Learning (RL) tuning is adopted to obtain the LLM dedicated for Proofread feature. Results suggested that our rewrite task tuning and reinforcement learning recipe significantly improves the proofreading performance of the foundation models. To reduce the serving cost, we build our feature on top of the medium sized LLM PaLM2-XS, which could be fit int a single TPU v5 after 8-bit quantization. We further optimize latency with bucket keys, segmentation and speculative decoding (Leviathan et al., 2023). Our model now is launched to benefit thousands of users with Pixel 8 devices.

Figure 1 exhibits our model quality on one extreme corrupted case from Andrej Karpathy[1], which indicates our tuned model is strong enough to handle various of heavy typo errors made by users.

The contribution of this paper can be summarized as follows:

- We propose the **Proofread** feature supported by the high quality LLM to boost the user typing experiences of Gboard. We finally launched the feature to real users with Pixel 8 devices, thousands of users benefit from it daily.
- We design and implement the whole system from data generation, metrics design to model tuning and deployment.
- We obtain a high quality model with cautiously synthetic data generation, multiple phased supervised fine-tuning and RL tuning. Specifically, we propose the Global Reward and Direct Reward in RL tuning stage, which improve the model significantly. Results shows that RL tuning could help reduce the grammar error significantly and thus the Bad ratio of PaLM2-XS model is reduced by 5.74% relatively.
- We deploy the model to TPU v5 in Cloud with highly optimized latency acquired by quantization, buckets, input segmentation and speculative decoding. Our results suggested that speculative decoding reduced the median latency by 39.4%.

2 Related Work

2.1 Controllable Text Generation

Controllable text generation using transformer-based pre-trained language models has become a rapid growing yet challenging new research hotspot (Zhang et al., 2023). Proofread falls into this scope with the requirement of modifying the input to fix the grammar errors without changing the original intention in the corrupted text.

Lots of applications could inherit from controllable text generation. Shu et al. (2023); Zhu et al. (2023) focus on text rewrite tasks, including paraphrasing (Xu et al., 2012; Siddique et al., 2020), style transfer (Riley et al., 2020; Zhang et al., 2020; Reif et al., 2021) and sentence fusion (Mallinson et al., 2022) and so on. Similarly, Text editing (Malmi et al., 2022) task also covers a wide range of sub-tasks such as paraphrasing, style transfer, spelling and grammatical error correction (Napoles et al., 2017), formalization (Rao and Tetreault, 2018), simplification (Xu et al., 2016) and elaboration (Logan IV et al., 2021).

Unlike these mentioned works, our paper only addresses a single application – Proofread but provides systematic approaches that optimize the model from different perspective such as quality,

[1]https://twitter.com/karpathy/status/1725553780878647482

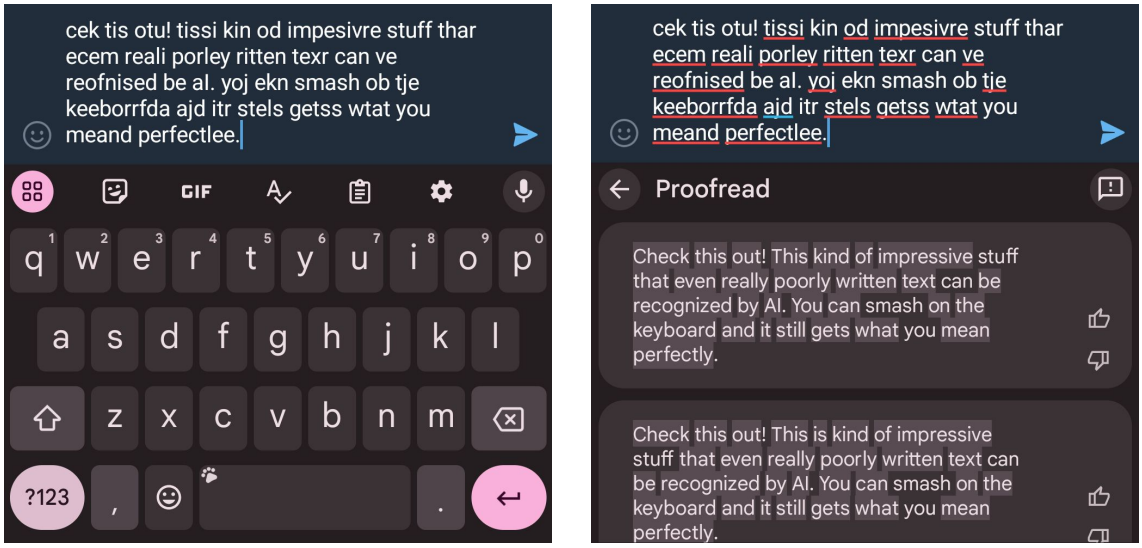

Figure 1: Proofread demo on a heavy corrupted text, the feature is triggered by clicking the "A" button in the left figure.

latency and resource usage.

2.2 Grammatical Error Correction (GEC)

Proofread falls into the area of GEC. Bryant et al. (2023) offers a comprehensive survey of the history and the current state of GEC. Specifically, before LLM, the popular solutions of GEC are edit-based approaches which corrections are applied on a sequence labelling (Omelianchuk et al., 2020) or sequence-to-sequence basis (Stahlberg and Kumar, 2020).

The recent studies to apply LLM to GEC mainly focus on prompting the LLM rather than supervised fine-tuning. Wu et al. (2023) compares ChatGPT to Grammarly, Coyne et al. (2023) compares GPT-3.5 and GPT-4 to two GEC system on English benchmarks. Davis et al. (2024) conducts a more comprehensive study by evaluating seven open-source and three commercial LLMs on four established GEC benchmarks.

Following the LLM trend, our system is built upon latest LLM backbone. But we apply instruction tuning approach to customize the LLM.

2.3 Instruction Tuning(IT)

Instruction tuning has been proven to be an efficient approach to boost model performance and generalization to unseen tasks (Chung et al., 2022; Sanh et al., 2021). Reinforcement learning with human feedback (RLHF) is leveraged to further extend instruction tuning in InstructGPT (Ouyang et al., 2022). Reinforcement learning with AI feedback (RLAIF) (Bai et al., 2022) could alleviate the heavy human preference data dependency, Zhu et al. (2023); Cheng et al. (2021) further replace the reward model in RLAIF with a heuristic model, which will be adopted in this paper to boost the quality. Our instruction tuning approach is inspired by the previous works and also follows the 2-step tuning process. We designed the synthetic data generation and RL strategy in a heuristic way that favors the proofreading task.

2.4 Latency Optimization

Numerous techniques aim to speed up inference of LLMs, which can be categorized into two major lines according to the focus point. The first line mainly focuses on model or algorithm side, including model compression with pruning (Xia et al., 2023b) and sparsity (Xia et al., 2023a), quantization (Dettmers et al., 2022), small model design (Timiryasov and Tastet, 2023; Liu et al., 2024), attention computation optimizations like low-rank approximation (Katharopoulos et al., 2020), sparse attention (Roy et al., 2021) etc.

The other line explores acceleration along with hardwares, exemplified works includes FlashAttention (Dao et al., 2022), FlexGen (Sheng et al., 2023), which considers hardware scheduling and weight movement, and speculative decoding (Leviathan et al., 2023; Chen et al., 2023), which leverages the parallism of hardwares to pre-conduct computation with sampling method.

We adopt quantization and speculative decoding to accelerate the inference speed in the model deployment.

3 Dataset

The upper half of Figure 2 illustrates the pipeline to generate the dataset. We initially sample data from the web crawled dataset, which is then processed by a GEC model to fix the grammar errors. Each item in the dataset consists of a source sentence with several possible reference sentences.

Grammar errors are then synthesized into the source sentence to simulate users' inputs, various kinds of errors which frequently happen in Gboard real scenarios are involved in this step, including:

- character omission, e.g., "hello" as "hllo"
- character insertion, e.g., "hello" as "hpello"
- transposition, e.g., "hello" as "hlelo"
- double tap, e.g., "hello" as "heello"
- omit double characters, e.g., "hello" as "helo"
- Gaussian-based positional errors, e.g., "hello" as "jello"

To align the dataset with real use cases, the data with synthetic errors are then passed to the Gboard simulator to fix errors by leveraging Gboard's built-in literal decoding, KC and AC functions. Moreover, several heuristic rules were then applied to fix cases such as emoji/emoticons alignment, date time formatting, and URL patterns.

The last step is to filter the noise data by utilizing LLM with careful designed instructions to avoid polluting the model. Data is diagnosed by various dimensions, including:

- The reference sentence still has errors remaining.
- The reference sentence itself is not fluent or clear enough.
- The reference sentence has different meaning as the source sentence.
- The reference sentence has different tones, aspects and tense from the source sentence.

To maximally benefit the model quality, the criteria above coordinates to the metrics defined in the following section.

An example of the synthetic dataset is showcased below:

> **Source**: "Good Moning! hey si, how. a u dou?"
> **Reference1**: "Good morning! Hey sir, how are you doing?"
> **Reference2**: "Good morning! Hey sister, how are you doing?"

Moreover, part of the examples labeled by human rater are used as the golden set for evaluation.

4 Metrics

It's of key importance to define the correct metrics which are aligned to user experiences online before the feature goes to public. In this section, several metrics are designed to measure the model quality.

Given the three elements, input (corrupted text), answer (predicted candidate from the model) and target (ground truth), we present the following metrics.

- **EM** / Exact Match Ratio: ratio of answer equal to target exactly.
- **NEM** / Normalized Exact Match Ratio : ratio of answer equal to target ignoring capitalization and punctuation.
- **Error** Ratio: ratio of answer containing grammar errors, which is conducted by LLM with specific instruction.
- **Diff** Meaning Ratio: ratio of answer and target don't have the same meaning, which is also conducted by LLM with specific instruction.
- **Good** Ratio: ratio of answer without grammar error and has the same meaning with target.
- **Bad** Ratio: ratio of answer either have grammar error or has different meaning with target.

From the definition, the Good/Bad ratios combining Error check and Diff Meaning check, are the primary metrics due to their robustness from LLM. The bad ratio is a bit more important as it portrays how much the users could tolerate the errors made by model. The combination of Error / Diff Meaning checks is also leveraged as the reward in RL phase of model tuning. EM/NEM ratios are referenced as supporting indicators as they are too strict for examples with multiple references.

5 Model tuning

The lower half of Figure 2 illustrates the tuning steps of the model for Proofread. We start from instruction-tuned models. PaLM2-XS model from Anil et al. (2023) is the candidate model.

5.1 Supervised Fine-tuning

The initial step after choosing the checkpoint is to fine-tune the model on the rewrite dataset, which contains hundreds of text rewriting tasks from Shu et al. (2023); Zhu et al. (2023). We assume that fine-tuning on similar tasks is beneficial to the final quality of Proofread. After that, the models are fine-tuned on synthetic dataset.

The evaluation results of multiple phased tuning are displayed in Table 1. It's natural to observe

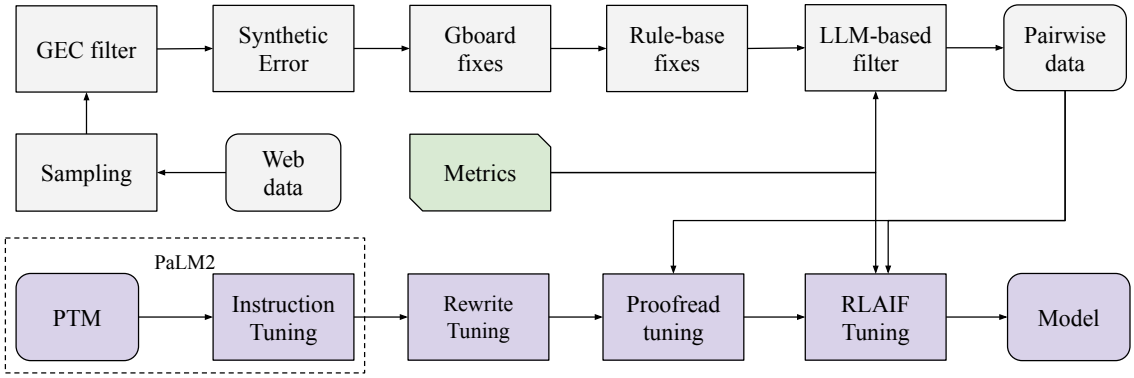

Figure 2: Data synthesis and Model tuning pipeline

that after supervised fine-tuning on the synthetic dataset, the model quality can be largely improved from 65.48% to 83.80%. An interesting finding is that though fine-tuning on Rewrite dataset degrades the quality, sequential fine-tuning on Rewrite and Proofread datasets yields the best results with Good ratio 84.68% and Bad ratio 15.32%, which we argue the robustness is enhanced by the large size of the combined dataset, by comparing M2 and M3, we can further conclude that Rewrite tuning contributes to the intent preservation.

5.2 Reinforcement Learning

RLAIF is leveraged with heuristic rewards in our model tuning following Zhu et al. (2023) to avoid relying on human labelers. Two alternative heuristic rewards based on LLM are designed in this paper.

- **Global Reward**: With few-shot examples, the LLM tells whether a candidate is a good fix of the corrupted inputs.
- **Direct Reward**: As the goal is to improve the Good Ratio, we directly convert the grammar error check and diff meaning check into rewards, both relying on LLM and will be combined as the final reward. This requires the ground truth included in the example.

Proximal Policy Optimization (PPO) (Schulman et al., 2017) is facilitated to optimize the model. KL divergence is involved to help model keep the ability to recover the original text (Peters et al., 2010; Mitchell et al., 2023).

The second part of Table 1 exhibits the results of RL tuning with different rewards. It's observed that the Bad ratio of PaLm2-XS model could be improved by 3.65% and 5.74% relatively through applying the RL with Global Reward and RL with Direct Reward respectively.

Specifically, RL excels at reducing the grammatical error but struggles to maintain meaning alignment between prediction and ground truth by comparing M3, M4 and M5. We argue that the optimizing meaning is inherently more subjective and complex comparing than grammar. Additionally, RL reduces the EM and NEM ratios, indicating a shift in the output distribution for both correct and incorrect cases. While increasing the KL divergence penalty can mitigate this (See M5 and M6), it doesn't significantly improve the Good/Bad ratios. We suspect the defined metrics might have inherent conflicts. Future work will reply on online metrics and real user data to drive further improvement.

6 Model Serving

Google's TPUv5e (Google, 2023) is utilized to serve the Proofread model, which is the latest Google TPU chip with 16GB HBM. 8-bit quantization is facilitated to reduce the memory footprint and latency without observing quality degradation.

In the context of our research, which predominantly focuses on deployment within chat applications, it has been observed that the average sentence length seldom exceeds 20 words. Consequently, we have established a discrete set of bucket keys, specifically [16, 32, 64, 128], to categorize the input data accordingly. Furthermore, we have calibrated the temperature parameter to a value of 0.3, aiming to maintain a constrained level of creativity in the proofreading outcomes, thereby ensuring relevance and coherence.

To be capable of handling more extensive documents, a systematic approach is employed wherein the document is segmented into individual paragraphs. All paragraphs are then processed in parallel, allowing for a more manageable and efficient analysis.

Table 1: The metrics of PaLM2-XS tuned on various phases on the Golden dataset. The upper half focuses on the supervised fine-tuning, and model variants in Reinforcement Learning phase are listed in the lower half.

Model ID	PaLM variant	EM(%)	NEM(%)	Good(%)	Bad(%)	DIFF(%)	ERROR(%)
M0	PALM2-XS	29.96	45.80	65.48	34.52	18.56	30.32
M1	M0 + Rewrite	23.44	40.90	59.48	40.52	19.04	37.04
M2	M0 + Proofread	37.88	55.30	83.80	16.20	12.08	8.12
M3	M0 + Rewrite + Proofread	39.16	56.20	84.68	15.32	10.60	9.68
M4	M3 + RL Global Reward	35.92	53.80	85.24	14.76	11.12	6.80
M5	M3 + RL Direct Reward	32.20	50.20	**85.56**	14.44	11.52	5.68
M6	M5 + Large weight on KL	39.08	55.40	84.76	15.24	10.96	8.88

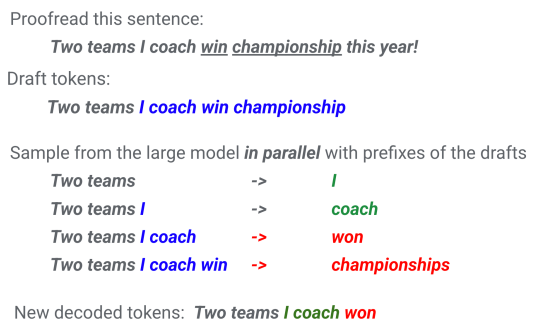

Figure 3: Example of the speculative decoding process for the proofreading task. The words in green color are selected draft by the LLM. This process speeds up the decoding without quality regression.

Table 2: Latency improvement with speculative decoding.

Decoding	Latency (ms)
Baseline	314.4
+ speculative	190.6 (-39.4%)

Additionally, our methodology incorporates the use of speculative decoding (Leviathan et al., 2023), complemented by heuristic drafter models that are tailored to align with user history patterns. Under our proofreading case, the initial input would naturally fit into the speculative draft so external drafter models are needed. We share an example to illustrate the process in Figure 3. This innovative approach significantly contributes to the reduction of operational costs. Through empirical evaluation in Table 2, we have recorded a 39.4% reduction on median latency per serving request, as measured on Tensor Processing Unit (TPU) cycles, underscoring the efficiency of our system in real-time applications.

7 Conclusions

This paper presents a novel Proofread feature implemented within Gboard, powered by a carefully refined LLM. Our work demonstrates the significant potential of LLMs to enhance the users' typing experiences by providing high-quality sentence- and paragraph-level corrections. We detailed our comprehensive approach, encompassing synthetic data generation pipeline aligned with real-world use cases, multifaceted metrics design, two-stage model tuning (multiple phased SFT followed by RL) and the efficient model deployment.

Specifically, our findings reveal that rewrite task tuning benefited the SFT model by enhancing the meaning alignment ability of the model. Additionally, we discovered the value of global and direct rewards during RL tuning, which could further improve the model by reduce grammar errors significantly. Rigorous experiments demonstrated that our tuned PaLM2-XS model achieved an impressive 85.56% good ratio and 14.44% bad ratio. The successful deployment of the model on TPU v5, leveraging optimizations such as quantization, bucket inference and speculative decoding, highlights its real-world viability.

This work underscores the transformative power of LLMs in the realm of user input experiences. Future research directions include leveraging real-user data, multilingual adaption, personalized assistance for diverse writing styles and privacy-preserving on-device solutions. This technology has the potential to fundamentally improve how we interact with our devices.

Acknowledgments

The authors would like to thank Ananda Theertha Suresh, Jae Ro, Ziteng Sun, Shankar Kumar, Lan Wei, Yuki Zhong, Zhe Su, JinDong Chen for their insightful discussions and support.

References

Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. 2023. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*.

Shiri Azenkot and Shumin Zhai. 2012. Touch behavior with different postures on soft smartphone keyboards. In *Proceedings of the 14th international conference on Human-computer interaction with mobile devices and services*, pages 251–260.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

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*, 49(3):643–701.

Charlie Chen, Sebastian Borgeaud, Geoffrey Irving, Jean-Baptiste Lespiau, Laurent Sifre, and John Jumper. 2023. Accelerating large language model decoding with speculative sampling. *arXiv preprint arXiv:2302.01318*.

Ching-An Cheng, Andrey Kolobov, and Adith Swaminathan. 2021. Heuristic-guided reinforcement learning. *Advances in Neural Information Processing Systems*, 34:13550–13563.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Yunxuan Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. *arXiv preprint arXiv:2210.11416*.

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. *arXiv preprint arXiv:2303.14342*.

Tri Dao, Dan Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. 2022. Flashattention: Fast and memory-efficient exact attention with io-awareness. *Advances in Neural Information Processing Systems*, 35:16344–16359.

Christopher Davis, Andrew Caines, Øistein 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. *arXiv preprint arXiv:2401.07702*.

Tim Dettmers, Mike Lewis, Younes Belkada, and Luke Zettlemoyer. 2022. Gpt3. int8 (): 8-bit matrix multiplication for transformers at scale. *Advances in Neural Information Processing Systems*, 35:30318–30332.

Google. 2023. Tpu system architecture. https://cloud.google.com/tpu/docs/system-architecture-tpu-vm#tpu_v5e. Accessed: 2024-03-15.

Angelos Katharopoulos, Apoorv Vyas, Nikolaos Pappas, and François Fleuret. 2020. Transformers are rnns: Fast autoregressive transformers with linear attention. In *International conference on machine learning*, pages 5156–5165. PMLR.

Yaniv Leviathan, Matan Kalman, and Yossi Matias. 2023. Fast inference from transformers via speculative decoding.

Zechun Liu, Changsheng Zhao, Forrest Iandola, Chen Lai, Yuandong Tian, Igor Fedorov, Yunyang Xiong, Ernie Chang, Yangyang Shi, Raghuraman Krishnamoorthi, et al. 2024. Mobilellm: Optimizing sub-billion parameter language models for on-device use cases. *arXiv preprint arXiv:2402.14905*.

Robert L Logan IV, Alexandre Passos, Sameer Singh, and Ming-Wei Chang. 2021. Fruit: Faithfully reflecting updated information in text. *arXiv preprint arXiv:2112.08634*.

Jonathan Mallinson, Jakub Adamek, Eric Malmi, and Aliaksei Severyn. 2022. Edit5: Semi-autoregressive text-editing with t5 warm-start. *arXiv preprint arXiv:2205.12209*.

Eric Malmi, Yue Dong, Jonathan Mallinson, Aleksandr Chuklin, Jakub Adamek, Daniil Mirylenka, Felix Stahlberg, Sebastian Krause, Shankar Kumar, and Aliaksei Severyn. 2022. Text generation with text-editing models. *arXiv preprint arXiv:2206.07043*.

Eric Mitchell, Rafael Rafailov, Archit Sharma, Chelsea Finn, and Christopher D Manning. 2023. An emulator for fine-tuning large language models using small language models. *arXiv preprint arXiv:2310.12962*.

Courtney Napoles, Keisuke Sakaguchi, and Joel Tetreault. 2017. Jfleg: A fluency corpus and benchmark for grammatical error correction. *arXiv preprint arXiv:1702.04066*.

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

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. 2022. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Tom Ouyang, David Rybach, Françoise Beaufays, and Michael Riley. 2017. Mobile keyboard input decoding with finite-state transducers. *arXiv preprint arXiv:1704.03987*.

Jan Peters, Katharina Mulling, and Yasemin Altun. 2010. Relative entropy policy search. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 24, pages 1607–1612.

Sudha Rao and Joel Tetreault. 2018. Dear sir or madam, may i introduce the gyafc dataset: Corpus, benchmarks and metrics for formality style transfer. *arXiv preprint arXiv:1803.06535*.

Emily Reif, Daphne Ippolito, Ann Yuan, Andy Coenen, Chris Callison-Burch, and Jason Wei. 2021. A recipe for arbitrary text style transfer with large language models. *arXiv preprint arXiv:2109.03910*.

Parker Riley, Noah Constant, Mandy Guo, Girish Kumar, David Uthus, and Zarana Parekh. 2020. Textsettr: Few-shot text style extraction and tunable targeted restyling. *arXiv preprint arXiv:2010.03802*.

Aurko Roy, Mohammad Saffar, Ashish Vaswani, and David Grangier. 2021. Efficient content-based sparse attention with routing transformers. *Transactions of the Association for Computational Linguistics*, 9:53–68.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, et al. 2021. Multitask prompted training enables zero-shot task generalization. *arXiv preprint arXiv:2110.08207*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms. *arXiv preprint arXiv:1707.06347*.

Ying Sheng, Lianmin Zheng, Binhang Yuan, Zhuohan Li, Max Ryabinin, Beidi Chen, Percy Liang, Christopher Ré, Ion Stoica, and Ce Zhang. 2023. Flexgen: High-throughput generative inference of large language models with a single gpu. In *International Conference on Machine Learning*, pages 31094–31116. PMLR.

Lei Shu, Liangchen Luo, Jayakumar Hoskere, Yun Zhu, Canoee Liu, Simon Tong, Jindong Chen, and Lei Meng. 2023. Rewritelm: An instruction-tuned large language model for text rewriting. *arXiv preprint arXiv:2305.15685*.

AB Siddique, Samet Oymak, and Vagelis Hristidis. 2020. Unsupervised paraphrasing via deep reinforcement learning. In *Proceedings of the 26th ACM SIGKDD international conference on knowledge discovery & data mining*, pages 1800–1809.

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

Inar Timiryasov and Jean-Loup Tastet. 2023. Baby llama: knowledge distillation from an ensemble of teachers trained on a small dataset with no performance penalty. *arXiv preprint arXiv:2308.02019*.

Haoran Wu, Wenxuan Wang, Yuxuan Wan, Wenxiang Jiao, and Michael Lyu. 2023. Chatgpt or grammarly? evaluating chatgpt on grammatical error correction benchmark. *arXiv preprint arXiv:2303.13648*.

Haojun Xia, Zhen Zheng, Yuchao Li, Donglin Zhuang, Zhongzhu Zhou, Xiafei Qiu, Yong Li, Wei Lin, and Shuaiwen Leon Song. 2023a. Flash-llm: Enabling cost-effective and highly-efficient large generative model inference with unstructured sparsity. *arXiv preprint arXiv:2309.10285*.

Mengzhou Xia, Tianyu Gao, Zhiyuan Zeng, and Danqi Chen. 2023b. Sheared llama: Accelerating language model pre-training via structured pruning. *arXiv preprint arXiv:2310.06694*.

Wei Xu, Courtney Napoles, Ellie Pavlick, Quanze Chen, and Chris Callison-Burch. 2016. Optimizing statistical machine translation for text simplification. *Transactions of the Association for Computational Linguistics*, 4:401–415.

Wei Xu, Alan Ritter, William B Dolan, Ralph Grishman, and Colin Cherry. 2012. Paraphrasing for style. In *Proceedings of COLING 2012*, pages 2899–2914.

Hanqing Zhang, Haolin Song, Shaoyu Li, Ming Zhou, and Dawei Song. 2023. A survey of controllable text generation using transformer-based pre-trained language models. *ACM Computing Surveys*, 56(3):1–37.

Yi Zhang, Tao Ge, and Xu Sun. 2020. Parallel data augmentation for formality style transfer. *arXiv preprint arXiv:2005.07522*.

Yun Zhu, Yinxiao Liu, Felix Stahlberg, Shankar Kumar, Yu-hui Chen, Liangchen Luo, Lei Shu, Renjie Liu, Jindong Chen, and Lei Meng. 2023. Towards an on-device agent for text rewriting. *arXiv preprint arXiv:2308.11807*.