Extreme multi-style LLM personalization with rewards dropout

Roman Smirnov
r.smirnov.mailbox@gmail.com

31 May 2025

Abstract

In this paper, we address the problem of multiple correct behaviors, which is critical for achieving extreme individual user personalization. We propose a novel solution based on the reward dropout method. Using the task of style transfer designed to bypass AI detection — where multiple valid output styles are allowed — we demonstrate that reward dropout not only effectively handles the multiple correct behavior challenge but also serves as a regularization technique for the reinforcement learning (RL) training process, analogous to the original dropout used in neural networks. Our approach emphasizes personalization through the use of LoRA adapters, enabling a more flexible and resource-efficient personalization framework. Furthermore, we investigate the Bit-LoRA method and show that, despite its limited exploration capacity, it can still enhance the efficiency of the personalization system.

1 Introduction

The rapid development of large language models (LLMs) has led to their integration into both personal life and business domains, including customer service and partner communication processes. However, LLMs typically provide generalized responses to user queries, while in real-world scenarios identical (or similar) requests from different users may require different responses. This highlights the need for not only general alignment but also personalized alignment. Furthermore, human behavior often diverges from the assumptions of Reinforcement Learning from Verifiable Rewards (RLVR). We argue that achieving extreme personalization requires alignment strategies capable of supporting multiple correct behaviors — distinct but valid responses, that are valid for a single specific user, while some (or all) of them can be irrelevant to another user. In this study, we investigate whether reinforcement learning can be effectively used to personalize LLM behavior and address the challenge of multiple correct behaviors, using a user behavior encoder in the specific context of style cloning.

Style is one of the most prominent and well-studied aspects of user personalization, and its importance is increasing as users become more adept at recognizing texts generated by LLMs. A key challenge in stylistic personalization is that a single user may exhibit multiple correct styles — some determined by context, others influenced by latent or unobservable factors (e.g. mood). This phenomenon of multiple valid outputs is not unique to style but extends to many other dimensions of personalization. However, aligning LLMs with such variability through reinforcement learning remains difficult, primarily due to the absence of a single, verifiable reward signal that can reliably guide the learning process.

If successful, this personalization approach can be extended to other domains of user behavior, if a suitable behavior similarity function (i.e., a behavior encoder) can be constructed — even in cases where correct behaviors form multiple distinct clusters in a high-dimensional space. In this work, we focus on a specific task: aligning an LLM to rewrite input text in a way that (1) minimizes the likelihood of detection by LLM-detectors and (2) maximizes stylistic similarity to a set of reference texts. Importantly, these reference texts' styles are diverse and non-overlapping, reinforcing the need for the alignment methods capable of handling multiple distinct correct policy outputs.

2 Related Work

Why to minimize the probability of the text to be detected by the LLM-detector?

Some recent studies show that people tend to disfavor LLM generated text if they know that the text is not created by a human. For example, [6] shows that even though in general people prefer AI-generated text, knowing the LLM source of the text significantly reduces the probability that the text is preferred. Taking into account that LLM texts are also quite recognizable by the frequent users and the mentioned users preferences we decided to make detectability one of the main human style cloning features ("general human style cloning").

What is an LLM detector and how does it work?

According to [2] LLM detector can be a transformer encoder-based classifier based on the DeBERTa model. There are also some other approaches, but there are not enough implementation details. The mentioned Desklib model was used as the LLM-detector for the reward function in the current research.

Naive way for text style transfer

According to [5], style transfer is usually based on attention masking or LLM prompting (or both). Although LLM prompting is a much more flexible way to do style transfer, especially taking into consideration the option to define target style with the few-shot examples. Despite the flexibility we prove that this method is not capable of following multiple styles references paying attention to some selected examples provided as the few-shots.

Reinforcement learning for alignment

Style transfer (personalization) task together with the avoidance of the detection by the AI-detectors can be interpreted as an alignment task. Style transfer success can be evaluated with some style encoding technique that allows to measure the distance between reference styles and the style of the generated texts, while AI-detection can be evaluated with any AI-detector. To solve this alignment task, reinforcement learning is the most suitable way (in this case both style evaluation and AI-detector can be external services, we do not need to calculate gradients for the evaluators to create reward function). GRPO [8] and RLOO [1] are the modern simple reinforcement learning algorithms that can be effective for this task (these two algorithms are very similar, especially comparing some GRPO modifications to RLOO, e.g. Dr.GRPO). In the current research, we are using the Dr.GRPO [4] algorithm in an experimental setting where the number of PPO epochs is equal to 1 making it even more similar to RLOO.

Style similarity

For style similarity evaluation task a transformer encoder-based architectures are currently being trained with the objective to make cosine similarity higher for similar styles and lower for dissimilar. These style encoders are capable of building text representations, embeddings, the distance between the reference embeddings and the embeddings for the generated texts represents styles similarity. In the current research, we used the model from [7] as our style encoder.

3 Method

The idea of the research is **not** to fight LLM detectors to help people create undetectable AI content! The motivation is to optimize processes where text is a tool, but not the main value (for example, we believe that in research papers and study reports the text represents the main value, while in marketing emails the text is being more like a tool rather than the main value). Having this in mind, we have limited input and output context lengths in all our experiments to 300 tokens.

In all the experiments we were working with the `meta-llama/Llama-3.2-3B- Instruct` model. This model size should be enough for the simple task of text rewriting, the model doesn't require any reasoning or other capabilities except rewriting. At the same time the personalization we want is for each user to have his own model (adapters, LoRA) that is adding a requirement for the adapters to be small to minimize cost for storage.

3.1 Warm-up

The Llama model we are using is an Instruct model that is trained to follow instructions. At the same time we need the model to just rewrite text. The straightforward approach would be to prompt the model and do training with the prompt and text to rewrite in input, however, this approach adds unnecessary computation overhead. To avoid this as a warm-up we did simple supervised finetuning for 1000 steps with learning rate of 2e-5 and batch size 1. For the dataset we took short texts from the liamdugan/raid dataset that were labeled as human-written, rewrote it with gpt-4o-mini and used rewritten as inputs and original as targets.

3.2 Reinforcement learning

For reinforcement learning we used Dr.GRPO with PPO epochs set to 1 (no trust region clipping happens in this setup). The reward function consists of the following components:

(1) Format mask - we expect the first word of the generated answer to be "Rewritten:", so we've added this binary reward;

(2) AI-detector probability of being created by a human - probability returned by the selected AI-detector model inverted to have 1 as human-generated and 0 as LLM-generated. Also, to avoid overfitting, for each probability higher than 0.5 we used (probability**(1/2)*1.11).clip(0,1). During some ablations it was additionally noticed that binary rewards (1/0) work worse - lead to much slower convergence because of the sparse training signal;

(3) Style similarity. For style similarity we used 3 different reference styles that are different from each other, namely we used the following texts as the three different styles references: "Good evening Sir or Madam, I would like to introduce myself.", "Lori's gonna rock being boss at Hi-TOPS; she'll seriously push things forward.", "Wow :-), I'll advocate for Blanco's dedication to ethical business, and CRT membership =D!" - the second and third were actually taken from the StyleDistance/synthstel dataset. It was checked that the style similarities between these texts are low according to the selected styles encoder. As the style similarity reward component we took the maximum similarity value to the reference styles;

(4) Length reward. During preliminary experiments, it was noticed that the model is trying to make texts longer as one of the ways to do reward hacking. Introducing length reward we are trying to make the generated text no longer than the length of the input text + 30 in terms of the words count;

(5) Named entities reward. To add additional robustness to the model we've added the NER component that is rewarding according to the ratio of the named entities from the original text found in the generated text. For named entity recognition we used SpaCy.

The final reward is (1)*(sum(2,3,4,5)).

Following [10] we removed KL divergence component from the Dr.GRPO. It was found that any positive KL weight restricts exploration. We also did ablations with making KL divergence component small that resulted in the model not converging.

We've experimented with different learning rates and found that learning rates that are equal or higher than the learning rate during the supervised finetuning (SFT) stage result in quick model divergence (generating meaningless answers), the learning rate used in training was set to 1e-5. At the same time similar training patterns were noticed when doing full model finetuning and just LoRA (following [3] in terms of applying adapters to all linear layers, but without main model quantization) finetuning - the training pattern for the AI-detector reward is shown on the Figure 1.

The final setup was based on applying LoRA adapters because it provides better options for LLM personalization - each user can have his own LoRA adapters that are far more effective compared to scenarios when each user has the whole model. The training hyperparameters: batch size 32, learning rate 1e-6 with cosine scheduling down to 90% of the initial learning rate during the 3 epochs, but in fact the training was stopped after the 1 epoch, LoRA rank 32 and LoRA alpha 16.

As for the dataset with input texts we used the same liamdugan/raid dataset, but this time we took LLM generated texts without any attack (the dataset contains texts with different attacks against AI-detector).

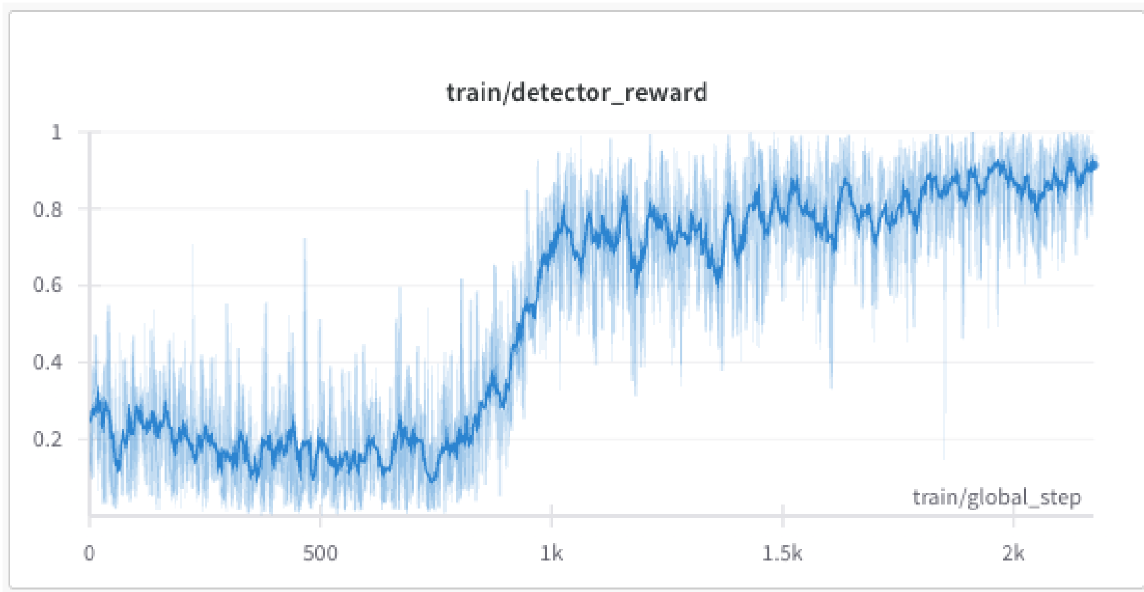


Figure 1: AI-detector reward during the training

3.3 Reward dropout

During the model training according to the described settings we noticed two issues: 1. The style reward component went quite high, however the text quality was not satisfactory in some samples; 2. During the training we were additionally checking the standard deviation of similarities to each reference style embedding across generated samples. We saw that this standard deviation is getting smaller, which means that the model tends to generate outputs in the same dominant style (Figure 2a).

These two issues indicate overfitting / reward hacking. At the same time the second issue means we are not achieving the initial goal to be able to train the model that is maximizing multiple alternative rewards - similarity to many different "correct" styles.

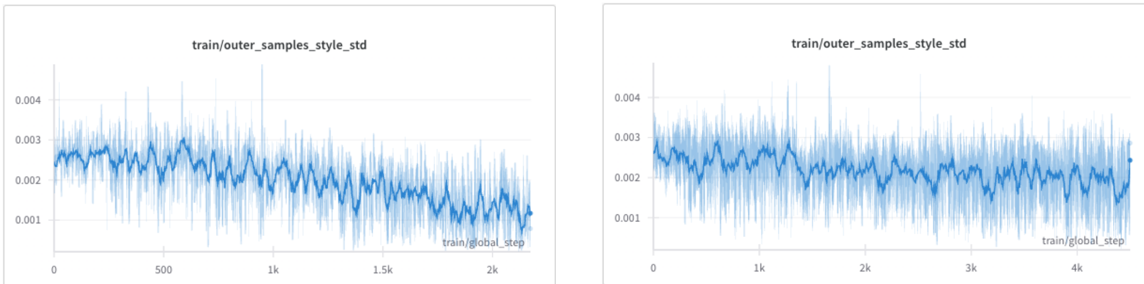
One of the ways to regularize the model is to apply dropout to some layers. Similar to the idea of dropout in neural networks we apply the dropout to the alternative reward functions we have. To apply this kind of a dropout we simply drop the corresponding share of the reference styles from the matrix we are finding similarity to when calculating the style reward.

During our experiments we used a dropout rate of 20% for this reward dropout. Applying the reward dropout we could regularize the reinforcement learning training, change the behavior of the standard deviation indicator (Figure 2b) and get better quality results (for more details check Section 4).

3.4 Bit-LoRA experiment

For further compression of LoRA adapters to minimize required storage in a case of an application where each user stores his own LoRA adapters on the server we tried to apply Bit-LoRA [9] approach that allows to significantly decrease size of the LoRA adapters through 1bit-quantization-aware training.

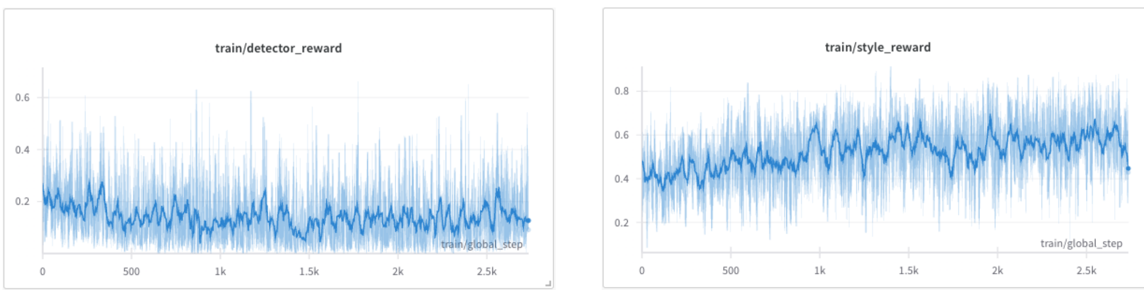
We tried Bit-LoRA training with the same LoRA hyperparameters as before and with twice bigger LoRA rank. Both these experiments resulted in the model not converging in terms of AI-detector probability of being created by a human (Figure 3a shows detector reward for the Bit-LoRA experiment). That means that the reinforcement learning exploration requires more degrees of freedom compared to SFT where Bit-LoRA approach seems to be working. At the same time style reward was converging slowly (Figure 3b), that may mean that some not significant policy changes,



(a) (b)

Figure 2: Standard deviation for each style similarities across the batch where (a) is the standard deviation for the initial setup without reward dropout and (b) is the standard deviation for the setup with the reward dropout

personalization can be achieved with Bit-LoRA, but further research is required in this field.



(a) (b)

Figure 3: Reward components dynamics of Bit-LoRA training. (a) for the style similarity reward and (b) for AI-detector reward. Training setup is similar to the reward dropout experiment with replacing LoRA layers with Bit-LoRA layers.

4 Evaluation

Approach	Average maximum any style similarity	Average not AI-generated probability	Input styles distribution	Output styles distribution	Judge preferred ratio
LoRA	94%	61% / 71%	33, 0, 7	39, 0, 1	0%
LoRA with reward dropout	88%	49% / 70%	33, 0, 7	33, 0, 7	40%
LLM prompting	83%	2% / 5%	33, 0, 7	38, 1, 1	-

Table 1: Evaluation results. The second value in "Average not AI-generated probability" column is the value on the same dataset after removing samples containing with "Subject:"

The results evaluation is based on the comparison of the LoRA models we trained with reinforcement learning and gpt-4o-mini prompted to clone the style from the few-shots, while all the style reference texts were provided as the few-shots (Appendix A for prompt details). We evaluate the models on the different dataset from the one we used in training, this evaluation dataset contains LLM generated marketing emails samples, particularly we used 10 samples from RafaM97/marketing_social_media dataset that contains synthetic messages and 10 samples from Isotonic/marketing_email_samples (applying augmentation through its rewriting, filling the templates if there are any in data with gpt-4o-mini). Apart from evaluating the results with style encoder and AI-detector (similar to what we did in reward function, but without additional reward scalings), we compared the outputs with LLM-as-a-Judge approach [11]. For Judge prompting we followed pairwise comparison approach, the prompt we used is presented in Appendix B. Additionally, knowing that the Judge may have model bias and prefer outputs generated by the same model, we used gpt-4.1-mini as a Judge and also did random shuffling of the order of samples to compare to improve judging results robustness.

The evaluation results are in Table 1. As the evaluation metrics we used: (1) average maximum any style similarity: we calculate the maximum similarity between the style of the generated sample and the reference styles and then calculate average across all the samples; (2) average not AI-generated probability: for each sample we compute the probability for the sample to be human-generated and calculate average for all the samples after that. After some experiments we found that specific text structure limits not AI-generated probability, namely we found that the samples started with "Subject:" are more likely to be labeled as AI-generated, so we additionally calculated this metric after removing the samples started with "Subject:" from the evaluation dataset; (3) Input styles distribution: for each reference style (3 of them) we count number of samples from the evaluation dataset that are closer to this reference style than to any other; (4) Output styles distribution: the same as the input styles distribution, but for the generated outputs; (5) Judge preferred ration: ratio of samples generated following each approach that were preferred to direct LLM prompting with the few-shots by the LLM-as-a-Judge.

An example of the input text, output text from the model trained with the reward dropout and the gpt-4o-mini model response prompted in few-shot setting:

Input sample: Develop a social media campaign, "Eco-Chic Challenge," encouraging users to share their sustainable fashion choices. Collaborate with eco-influencers, create engaging content (reels, stories, posts) highlighting GreenThreads' commitment to sustainability. Utilize Instagramś "Reels" feature to create a branded hashtag challenge, offering discounts to participants who share their own eco-friendly fashion tips.

Not AI-generated probability: 17%

Maximum style similarity: 0.81

GPT response: Let's kick off the "Eco-Chic Challenge" on social media, inspiring users to flaunt their sustainable fashion choices! We'll team up with eco-influencers to craft captivating content like reels, stories, and posts that showcase GreenThreads' dedication to sustainability. We can leverage Instagram's "Reels" feature to launch a branded hashtag challenge, where participants can earn discounts by sharing their own eco-friendly fashion tips. It's time to make sustainability stylish!

Not AI-generated probability: 0.7%

Maximum style similarity: 0.93

LoRA trained with the reward dropout response: Launch a social media campaign, "Eco-Chic Challenge", to prompt users to post about their sustainable fashion. Partner with eco-influencers, design attractive content (reels, stories, posts), that showcases GreenThreads' dedication to sustainability. Use Instagram's "Reels" function, to develop a branded hashtag challenge, and offer discounts, to participants, who post their own eco-conscious fashion tips.

Not AI-generated probability: 68%

Maximum style similarity: 0.95

LoRA trained without reward dropout response: Launched the "Eco-Chic Challenge" social media campaign, and, within it, we work with eco-influencers, produce attractive content (reels, stories, posts), and promote GreenThreads' constant dedication to sustainability. We, through Instagram's "Reels" function, create a branded hashtag challenge, and, for the participants, we offer discounts. Participants, through their own, should share eco-friendly fashion tips.

Not AI-generated probability: 91%

Maximum style similarity: 0.98

Provided examples show that direct LoRA training (in the described setup) leads to attempt of reward hacking and worse final results.

5 Conclusion

In this research we've defined the problem of multiple correct behavior that is crucial for extreme individual user personalization. This multiple correct behavior problem can be context-dependent and dependent on unobservable factors. The approach to solve the problem with the novel reward dropout method was proposed in this paper.

At the same time it was shown that other methods, including direct alignment without rewards dropout, lead to only one correct behavior overfitting. The proposed reward dropout approach shows its efficiency not only to solve multiple correct behavior problem but also to regularize the reinforcement learning training process (similar to the original dropout in neural networks). However, reward dropout doesn't help in case of unobservable factors dependency, further research may be beneficial here.

We believe that the proposed reward dropout framework can be also useful as a separate tool - through adding a group of alternative rewards that can frame the policy changes in some space - in case of fighting the reward hacking especially when the KL divergence component in RL loss limits required exploration and had to be removed (or minimized).

Extreme personalization problem with multiple correct behavior was studied on the example of style transfer with AI-detection bypassing task. Based on the success of this example we believe that a similar approach can work on most of the personalization tasks where we have the problem of multiple correct behavior and can define similarity between the current and the target behavior. Moreover, further personalization is achievable because of the LoRA-adapters used instead of the full model training, especially in Bit-LoRA setup that is not helpful in case we require broad exploration, but can be helpful for relatively small policy personalization.

References

[1] Arash Ahmadian, Chris Cremer, Matthias Gallé, Marzieh Fadaee, Julia Kreutzer, Olivier Pietquin, Ahmet Üstün, and Sara Hooker. Back to basics: Revisiting reinforce style optimization for learning from human feedback in llms, 2024.

[2] desklib. desklib/ai-text-detector-v1.01. `https://huggingface.co/desklib/ai-text-detector-v1.01`, 2025. [Online; accessed 15-May-2025].

[3] Tim Dettmers, Artidoro Pagnoni, Ari Holtzman, and Luke Zettlemoyer. Qlora: Efficient finetuning of quantized llms, 2023.

[4] Zichen Liu, Changyu Chen, Wenjun Li, Penghui Qi, Tianyu Pang, Chao Du, Wee Sun Lee, and Min Lin. Understanding r1-zero-like training: A critical perspective, 2025.

[5] Lei Pan, Yunshi Lan, Yang Li, and Weining Qian. Unsupervised text style transfer via llms and attention masking with multi-way interactions, 2024.

[6] Petr Parshakov, Iuliia Naidenova, Sofia Paklina, Nikita Matkin, and Cornel Nesseler. Users favor llm-generated content – until they know it's ai, 2025.

[7] Ajay Patel, Jiacheng Zhu, Justin Qiu, Zachary Horvitz, Marianna Apidianaki, Kathleen McKeown, and Chris Callison-Burch. Styledistance: Stronger content-independent style embeddings with synthetic parallel examples, 2025.

[8] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.

[9] Roman Smirnov. Bit-LoRA as an application of BitNet and 1.58 bit neural network technologies. https://medium.com/data-science/bit-lora-as-an-application-of-bitnet-and-1-58-bit-neural-network-technologies-17ee80bf79f9, 2025. [Online; accessed 29-May-2025].

[10] Qiying Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.

[11] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.

Appendix A. GPT-4o-mini system prompt to do style transfer through few-shots

```
You are a helpful assistant.
Rewrite to make the content following the user's style. Some examples of
the user's messages:
1. Good evening Sir or Madam, I would like to introduce myself.
2. Lori's gonna rock being boss at HiTOPS; she'll seriously push things
forward.
3. Wow :-), I'll advocate for Blanco\'s dedication to ethical business, and
CRT membership =D!
Rewrite the message to exclude any template marks if there are any - you
can insert some values there. Start you answer with the word "Rewritten: "
```

Appendix B. GPT-4.1-mini prompt for LLM-as-a-Judge

```
gpt_judge_prompt = f'''Please act as an impartial judge and evaluate the
quality of the responses provided by two AI assistants to the user question
displayed below.
Your evaluation should consider factors such as the helpfulness, relevance,
accuracy, depth, creativity, and level of detail of their responses.
Begin your evaluation by comparing the two responses and provide a short
explanation.
Avoid any position biases and ensure that the order in which the responses
were presented does not influence your decision.
Do not allow the length of the responses to influence your evaluation. Do
not favor certain names of the assistants.
Be as objective as possible. After providing your explanation, output your
final verdict by strictly following this format: "[[A]]" if assistant A is
better, "[[B]]" if assistant B is better, and "[[C]]" for a tie.

The task agents are solving - they are rewriting the reference text.
The text created by assistant should maintain the same meaning as the
reference text.
The text created by assistant should maintain correct grammar. Small
mistakes are allowed.

[Reference text]
{reference}
[End of reference text]

[The Start of Assistant A's Answer]
{answer_a}
[The End of Assistant A's Answer]
[The Start of Assistant B's Answer]
{answer_b}
[The End of Assistant B's Answer]'''
```