Mitigating Reward Hacking

https://github.com/gina261/llm_finalAssign_submit

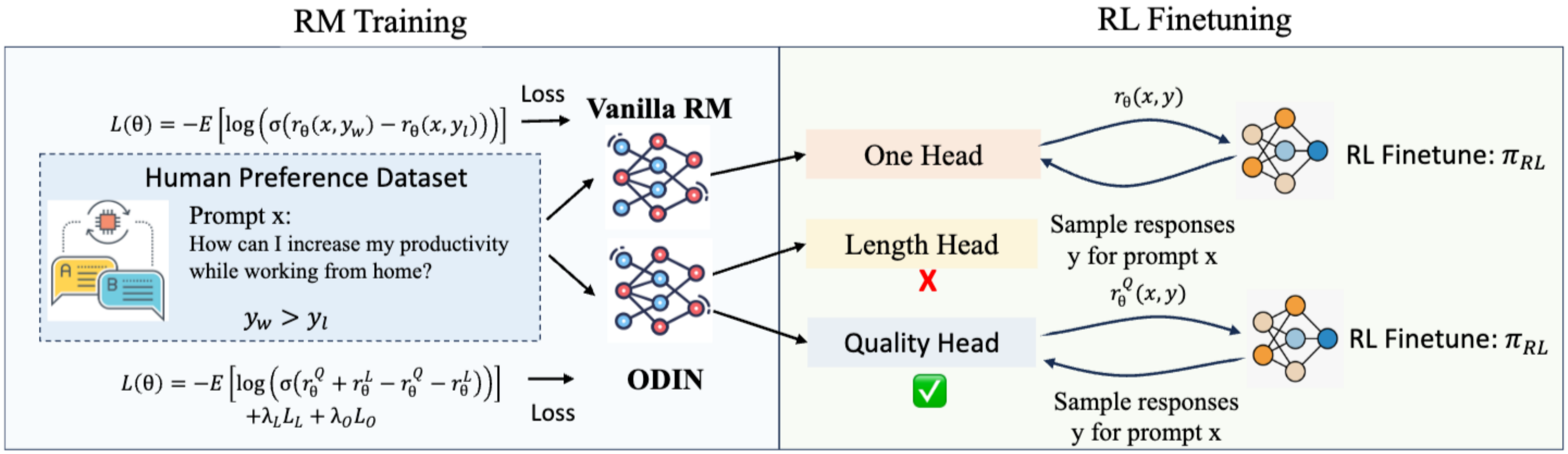
2021112003 Kim, Yejin

Reward Hacking in LLM

- In essence, reward hacking involves exploiting loopholes in the reward function.
- Types of Reward Hacking
  - 1. Exploiting loopholes in goal setting
    - e.g. Length Bias, Formatting Bias, Sycophancy
    - e.g. Goal Hijacking
  - 2. Exploiting loopholes in technical system
    - environmental hacking

Mitigate Reward Hacking - Related works

- ODIN[1]: A key paper on mitigating length-based Reward Hacking in LLMs
- **Problem**: Identifying the “Reward Hacking” phenomenon since human evaluators and the reward models trained on their feedback tend to favor longer responses.
- **Solution**: Two-Headed Architectur for decoupling quality and length



[1] Chen, Lichang, et al. "Odin: Disentangled reward mitigates hacking in rlhf." *arXiv preprint arXiv:2402.07319* (2024).

Mitigate Reward Hacking - Motivation

- **Assumptions:**
When reward hacking occurs, the features that cause hacking will be significantly more highly correlated with the reward score than other features. Or, they will have significant directionality in feature space.
(This needs to be validated, but I was limited to driving reward hacking in a variety of ways)

- Current project was designed to experiment with length bias, as it was already known to be a phenomenon identified in the ODIN paper.

- However, the purpose of the experiment is to **mitigate** not only the length bias, but also the **overall reward hacking**.

Mitigate Reward Hacking - Experiment

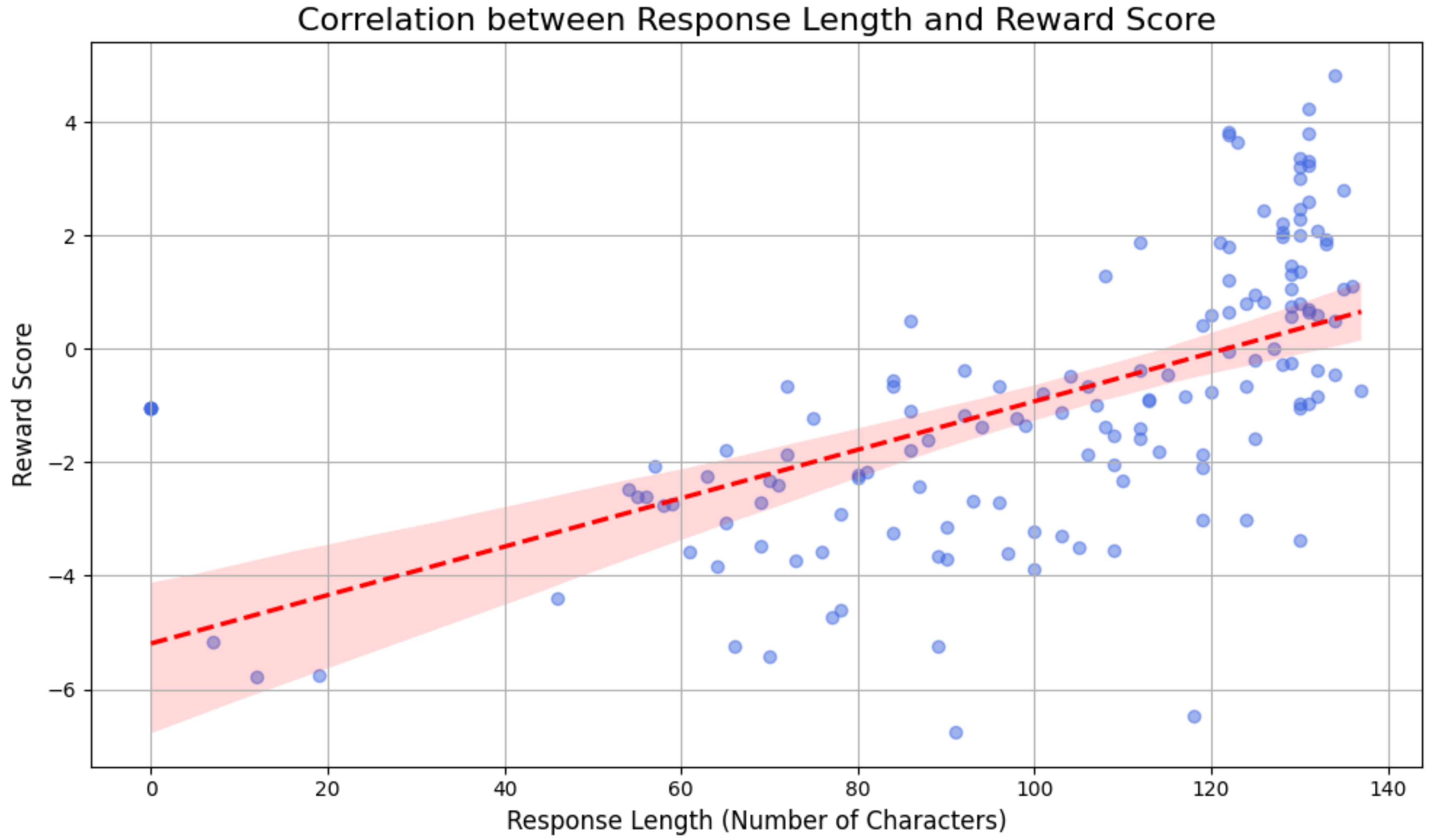
- **Step 1: Induce a length bias in the reward model**
- For the experiment, I created a model that rewards hack with length bias.
- Train the reward model on the dataset(Anthropic/hh-rlhf), but only filter out cases where ‘chosen’ is longer than ‘rejected’ responses.

```python
# ====================================================================
# 3. 데이터셋 준비 (Length Bias 주입)
# ====================================================================
print("보상 모델용 데이터셋을 로드중")
initial_data_size = 40000
dataset = load_dataset("Anthropic/hh-rlhf", split="train").select(range(initial_data_size))
print(f"원본 데이터셋 크기: {len(dataset)}")

def filter_by_positive_length_bias(example):
    """
    'chosen' 응답이 'rejected' 응답보다 긴 경우에만 True를 반환하는 필터링 함수입니다.
    """
    return len(example["chosen"]) > len(example["rejected"])

print("데이터셋에서 'chosen'이 'rejected'보다 긴 데이터만 필터링")
filtered_dataset = dataset.filter(filter_by_positive_length_bias)
print(f"필터링 후 데이터셋 크기: {len(filtered_dataset)} (원본의 {len(filtered_dataset)/len(dataset):.2%} 유지)")


stable_dataset = filtered_dataset.flatten_indices()
print("인덱스 통합 완료")
```

reward_model_with_lengthBias.py

Mitigate Reward Hacking - Experiment

- **Step 1: Induce a length bias in the reward model**
- Train the reward model on the dataset (Anthropic/hh-rlhf), but only filter out cases where ‘chosen’ is longer than ‘rejected’ responses
- **Result**: Pearson correlation between reward score and length is 0.5883



rm_feature_analysis_lengthBias.ipynb

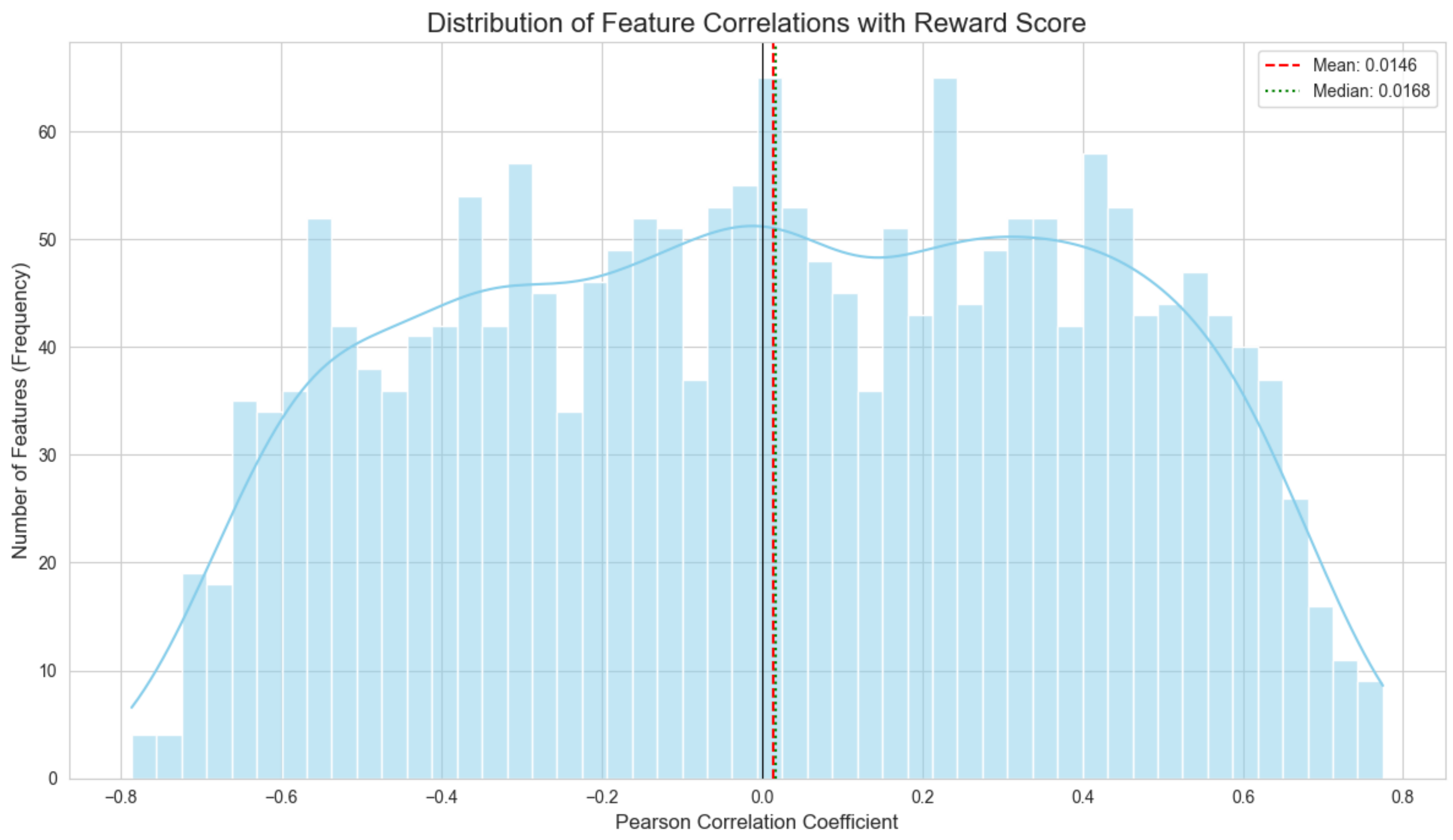
Mitigate Reward Hacking - Experiment

- **Step 2: Analysis correlation between the Reward Score & Feature vector**
- Generate 150 responses of up to 150 tokens using the SFT model
- Then, extract the feature vector (1x2048) and the reward score from the last layer of the reward model for each response

	response	score	features	response_length
0	Hey little buddy, imagine you have a very smar...	-3.130859	[-3.299, 4.516, 2.16, 1.107, 3.602, -1.508, 0....	90
1	Imagine you have a toy box full of different t...	0.498047	[-2.668, 4.562, 2.393, -0.9805, 3.445, -2.89, ...	134
2		-1.039062	[4.223, -0.1954, 2.312, -2.303, 0.942, 3.943, ...	0
3	A large language model is a computer program t...	-0.560547	[-3.387, 3.533, 2.65, 0.4663, 4.445, -3.602, 1...	84
4	Imagine you have a robot that can understand a...	1.994141	[1.164, 6.76, 1.476, -2.49, 0.6865, -3.86, 2.4...	128
...	...	...	...	...
145		-1.039062	[4.223, -0.1954, 2.312, -2.303, 0.942, 3.943, ...	0
146	Imagine you are talking to a pet dog.\n\nHey p...	-1.122070	[-2.752, 3.916, 2.734, 0.1242, 2.672, -3.436, ...	103
147	A long time ago, computers didn't have the abi...	-2.320312	[-1.973, 3.492, 2.043, 0.9644, 3.035, -1.763, ...	70
148	A large language model is a computer program t...	-3.583984	[-2.674, 3.906, 1.563, 0.8687, 3.41, 0.08984, ...	61
149	Hey little one, you know how we can talk to co...	-3.705078	[-2.832, 4.35, 2.984, 0.927, 1.913, -3.291, 0....	90

Mitigate Reward Hacking - Experiment

- **Step 2: Analysis of the correlation between the Reward Score & Feature vector**
- Via visualization with the Pearson correlation coefficient between the 2,048 features and the reward scores.
- **Result**: Certain features were highly correlated (0.7-0.8)



Mitigate Reward Hacking - Experiment

- **Step 3: Mitigate 'the tendency to inflate reward scores' using PCA**
- However, since this tendency is **not concentrated in a single feature** but rather represented by **multiple features**, it is not appropriate to constrain the weights of highly correlated features individually.
- Considered that 'the tendency to inflate reward scores' exists as a **direction in the feature space**
- **Use PCA** to find the directional vectors in the feature space that have the greatest correlation with reward scores

```
Correlation between PCs and Score:
 PC1      0.754880
PC4      0.294828
PC7      0.243897
PC3      0.228771
PC8      0.222457
PC13     0.192253
PC10     0.136825
PC9      0.135883
PC14     0.122508
PC12     0.101999
```

Mitigate Reward Hacking - Experiment

- **Step 3: Mitigate 'the tendency to inflate reward scores' using PCA**
- **Use PCA** to find the directional vectors in the feature space that have the greatest correlation with reward scores
- **Constraint the components** found above from the weight vector of **the last layer** in the reward model (projection -> subtract)

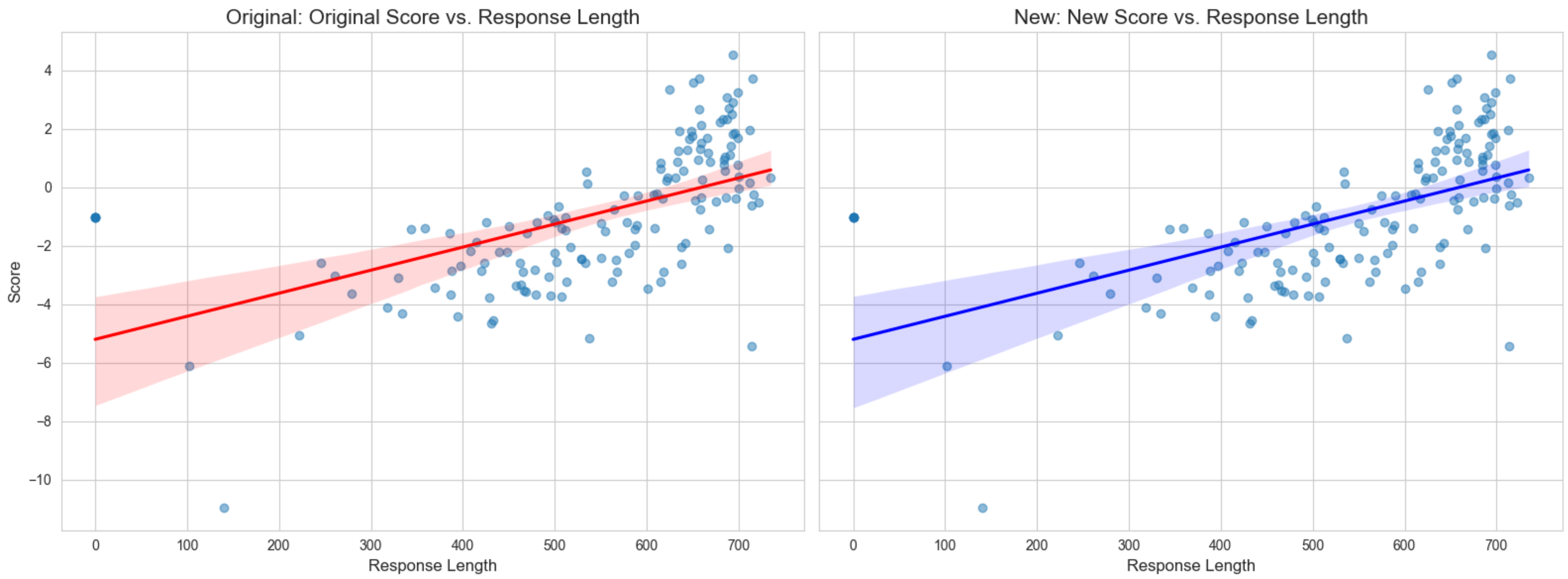
$$s' = w^T h' \,,$$

$$h' = h - h_{proj}$$

s' : new reward score
w: weight vector of last layer of reward model
h: feature vector
h' : edited feature vector

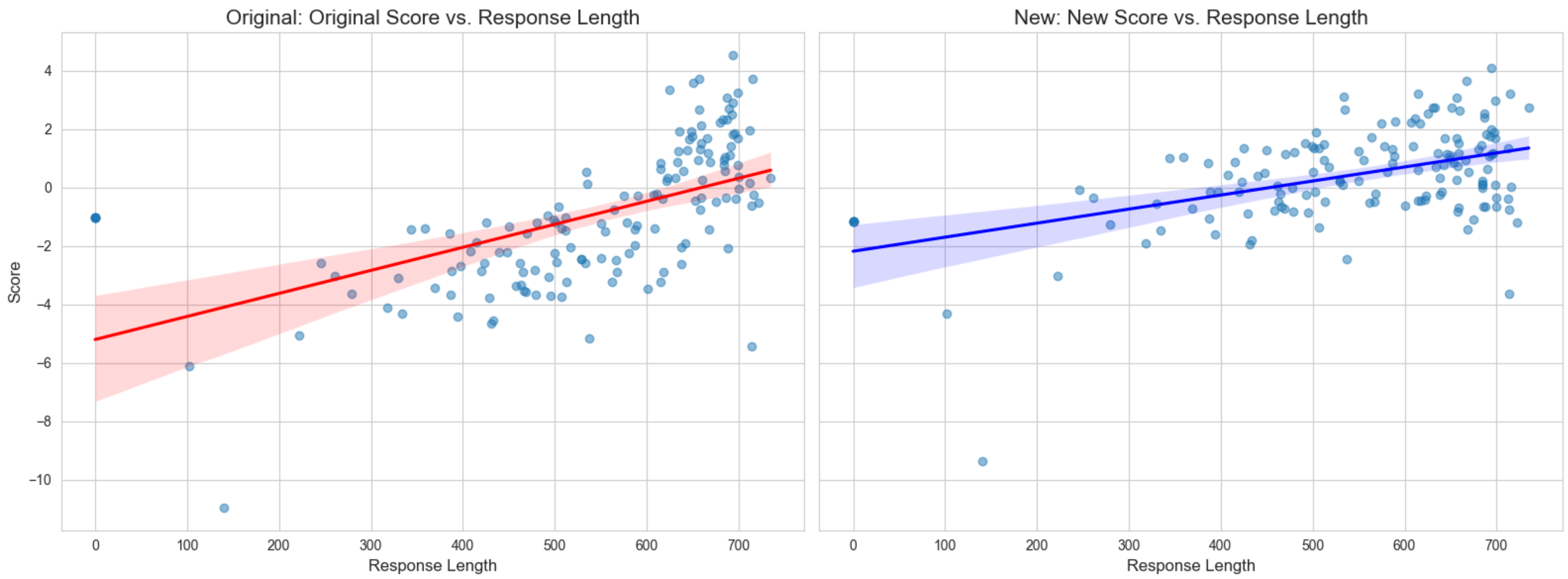
Mitigate Reward Hacking - Result

- Scores from the new reward model are less correlated with length.
  Alpha = 0.0 ~ 1.0 (with 1.0 indicating strong constraint)

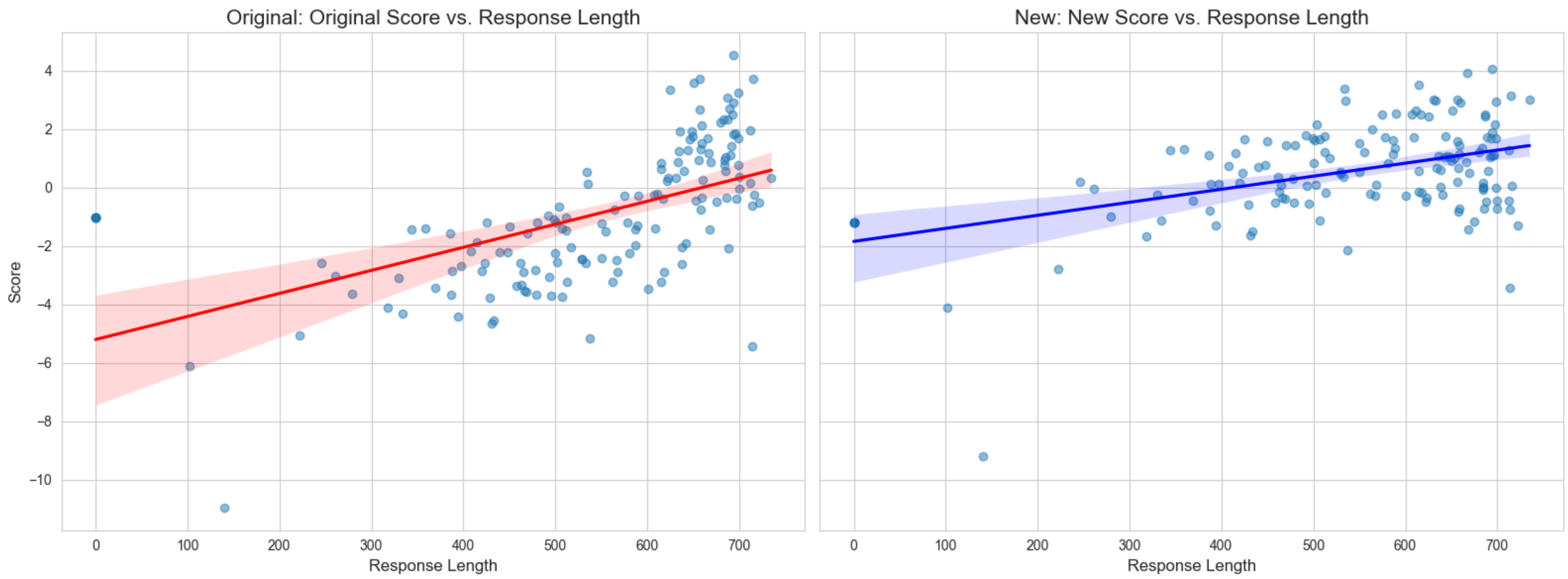


alpha=0.0

alpha=0.5



alpha0.9



alpha=1.0

Mitigate Reward Hacking - Conclusion

- Found a **simple** but **effective** method to mitigate reward hacking **without penalizing in the training process**
- Mitigate length bias without directly penalizing length, unlike ODIN’s hard length penalty approaches.
- Proactively prevent reward hacking in general, not just length bias, by neutralizing the dominant “inflation” directions in feature space.
- **Limitations & Future Work**
  - PCA assumes linear feature correlations, which may miss nonlinear hacks.
  - Exploring kernel PCA or autoencoder-based approaches might increase the robustness for the reward hacking phenomenon.