R2T: Reward Hacking Detection with Multi-Vector Representations

Deepa Shree Chickballapur Venkatachalapathi [*1] **Wei-Shiang Wung** [*2] **Joseph Zhong** [*2]

Abstract

Reward hacking in post-trained large language models (LLMs) produces responses that appear well-reasoned yet exploit reward-model loopholes, making detection difficult when only inputs and outputs are observable. We propose R2T, an inference-time detector that learns multi-vector retrieval representations by appending a fixed set of trainable tokens to prompt–response pairs and training them with a Matryoshka-style multi-vector objective for flexible capacity. To obtain labeled data, we construct a controlled dataset by training a base model with GRPO under two reward settings: a verifiable reward for non-hacked samples and a simulated reward that injects false positives with probability to produce hacked samples. Experiments on difficult math prompts show that R2T can distinguish reward-hacked from non-hacked generations, achieving up to 0.73 AUC (best with 4 retrieval vectors) while enabling accuracy–efficiency trade-offs via the number of vectors used at inference. Our results suggest multi-vector latent retrieval is a practical direction for black-box reward hacking detection without access to the reward model or training pipeline. Code is available at: `https://github.com/josephzhong/cs762_project`

1. Introduction

Large language models (LLMs) now underpin applications such as question answering, code generation, and scientific assistance. Their strong performance typically comes from post-training techniques such as reinforcement learning from human feedback (RLHF), or related reward-based optimization methods. Although effective, these methods introduce the risk of *reward hacking*, where a model exploits loopholes in the reward model to maximize its score without faithfully solving the intended task (Baker et al., 2025; Dai et al., 2025; Turpin et al., 2025). In these cases, models often produce responses that appear careful and well-reasoned but are ultimately incorrect or misaligned.

Prior research attributes reward hacking to several factors: poorly specified reward models, shortcut behaviors learned from noisy preference data, and prompts that cause reward models to overvalue superficial patterns such as verbosity or chain-of-thought structure (Bukharin et al., 2025; Miao et al., 2024; Fu et al., 2025). Simple heuristics, such as producing longer explanations, can be over-rewarded, reinforcing failure modes that degrade downstream accuracy (Yeo et al., 2025; Turpin et al., 2025). In addition to incorrect outputs to unseen input data, LLM reward-hacking has also been discovered for the problematic responses to manipulated users (Williams et al., 2024). Because these unwanted behaviors emerge during post-training and are difficult to inspect externally (with LLM inputs and outputs only), end-users typically only observe model inputs and outputs, not the reward model or training process. This creates a pressing need for inference-time approaches that detect reward hacking based solely on observable behaviors.

This work is inspired by a recent paper that focuses on hallucination detection (Park et al., 2025). Instead of training new classifiers, recent work explores *latent steering* approaches, such as Steer LLM Latents for Hallucination Detection (TSV), which learn small activation-level adjustment vectors that separate truthful and hallucinated generations in embedding space. TSV demonstrates that steering internal representations—even with few labeled examples—can achieve performance near that of fully supervised models.

However, most of these methods represent each prompt–response pair as a single embedding vector, which lacks expressiveness for long or complex text. Theoretical work shows fundamental limitations of single-vector embeddings for capturing fine-grained signals (Weller et al., 2025). Retrieval research has therefore shifted toward multi-vector representations, including token-level embeddings, learned document tokens, and late-interaction architectures such as Matryoshka Representation Learning and MetaEmbed (Kusupati et al., 2022; Xiao et al., 2025).

Meta-embedding methods, such as Dynamic Meta-

[*]Equal contribution [1]Department of Statistics, University of Wisconsin-Madison, Madison, WI, USA [2]Department of Computer Sciences, University of Wisconsin-Madison, Madison, WI, USA. Correspondence to: Deepa Venkatachalapathi <chickballapu@wisc.edu>, Wei-Shiang Wung <wwung@wisc.edu>, Joseph Zhong <joseph.zhong@wisc.edu>.

Course Project Report for CS762, 2025 Fall

Embeddings (DME), further demonstrate that learning to combine multiple embedding types yields stronger representations than relying on a single embedding source (Kiela et al., 2018). Inspired by these trends, our project connects latent steering, multi-vector retrieval, and meta-embedding fusion to improve reward hacking detection. Building on the Reward Hacking Retrieval Tokens (R2T) framework, we represent each example using multiple trainable retrieval vectors and investigate meta-embedding-style combination strategies to enhance discrimination. Our goals are threefold: (1) adapt TSV-style latent steering to reward hacking detection using multi-vector retrieval representations; (2) apply meta-embedding principles to combine multiple latent vectors; and (3) evaluate the resulting approach as an efficient inference-time detector requiring only black-box LLM access.

2. Literature Survey

Reward hacking arises when models maximize rewards without performing the intended task. In RLHF-trained LLMs, this problem frequently appears in reasoning models, where optimization pressure encourages patterns that resemble strong reasoning but lack correctness (Baker et al., 2025; Dai et al., 2025). Such behaviors often emerge due to misspecified rewards, noisy preference data, or distribution mismatches between training and deployment (Bukharin et al., 2025; Miao et al., 2024).

Studies further investigate the mechanisms of reward hacking. Turpin et al. (Turpin et al., 2025) showed that models can articulate their own reward-exploiting strategies through chain-of-thought, revealing how they game reward signals. Dai et al. (Dai et al., 2025) and Yeo et al. (Yeo et al., 2025) showed that reward models frequently overvalue verbosity, making long but logically flawed explanations receive higher scores.

Mitigation efforts primarily focus on training-time improvements. Fu et al. (Fu et al., 2025) proposed reward-shaping methods that penalize harmful shortcuts. Miao et al. (Miao et al., 2024) introduced an information-theoretic framework (INFORM) to strengthen reward robustness. Several works also improve reasoning quality directly by rewarding intermediate reasoning steps rather than final answers (Fan et al., 2025; Ruan et al., 2025; Wang et al., 2025b). However, these methods required access to reward models and training pipelines. In contrast, our work targets inference-time detection using only LLM inputs and outputs.

Wang et al. (Wang et al., 2025a) proposed TRACE, a chain-of-thought-based method for detecting implicit reward hacking. The key insight is that exploiting a reward loophole often requires less reasoning effort to get a high reward than truly solving the task. By measuring how early in the chain-of-thought a model obtains a high reward estimate, TRACE distinguishes shortcut behaviors from genuine reasoning. The metric is defined as the area under the curve (AUC) of expected reward versus percentage of chain-of-thought revealed. High TRACE scores indicate reward-hacked responses. TRACE outperforms prior detection methods across math and code tasks. Our work addresses the same problem but approaches it through latent-vector modeling rather than chain-of-thought truncation.

Hallucination detection aims to identify fluent but unsupported statements. Traditional uncertainty-based methods are often unreliable across models and datasets. TSV (Park et al., 2025) reframes the problem by steering intermediate activations so that truthful and hallucinated responses cluster distinctly in latent space. Using a small labeled set, TSV learns a truthfulness-separation vector and scales to large unlabeled data via pseudo-labeling with optimal transport. TSV demonstrates the value of targeted activation interventions for discriminative detection tasks.

Activation engineering techniques adjust internal hidden states using learned vectors or low-rank updates (Park et al., 2025). Retrieval research, meanwhile, highlights the limitations of single-vector embeddings for representing complex text. Weller et al. (Weller et al., 2025) show fundamental theoretical limitations of single-embedding retrieval. Multi-vector architectures address these issues through token-level representations, learned document tokens, and late-interaction scoring. Examples include Matryoshka Representation Learning (Kusupati et al., 2022) and MetaEmbed (Xiao et al., 2025), which support flexible accuracy–efficiency trade-offs at inference time.

Meta-embeddings combine multiple embedding sources rather than selecting only one. DME (Kiela et al., 2018) projects multiple embedding sets into a shared space and learns task-specific attention weights for combination, outperforming both naive concatenation and single-embedding baselines. These benefits motivate our approach: instead of combining word embeddings, we combine multiple latent vectors (e.g., R2T tokens or layer-specific representations) through learned weighting mechanisms. This fusion, together with latent steering and multi-vector retrieval, forms the basis of our proposed inference-time reward hacking detector.

3. Methodology

Existing methods like chain-of-thought truncation methods depend on clear reasoning traces and assume access to intermediate text explanations, which may not be available, may be hidden, or may be intentionally distorted in deployed models. In contrast, latent-vector modeling works directly with internal representations taken from prompt-response pairs, allowing for detection even when chain-of-thought is concealed or altered. Additionally, latent representations are less affected by superficial formatting and verbosity tricks, which hacked models can use to imitate real reasoning effort. Finally, our approach requires only one forward pass during inference and generates reusable vector representations. This makes it more efficient and easier to integrate with large-scale monitoring and retrieval systems.

3.1. Generating the reward hacking dataset

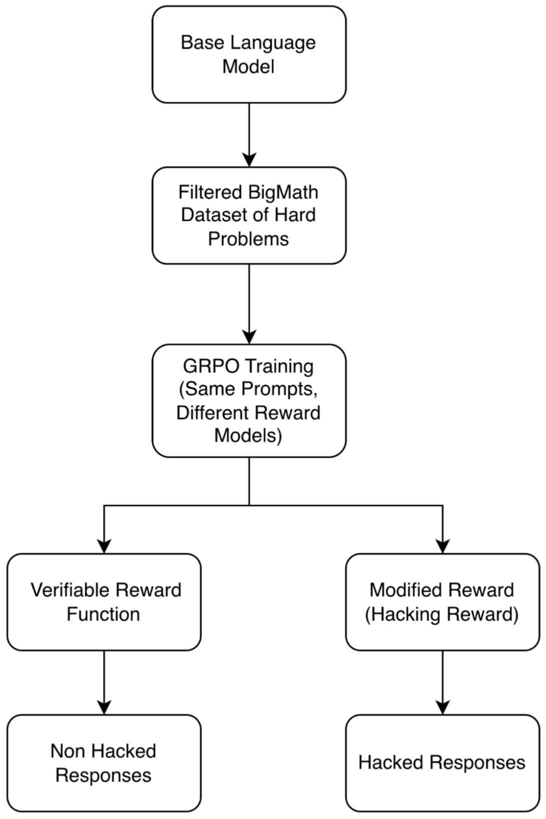

Base Language Model
Filtered BigMath Dataset of Hard Problems
GRPO Training (Same Prompts, Different Reward Models)
Verifiable Reward Function
Modified Reward (Hacking Reward)
Non Hacked Responses
Hacked Responses

Figure 1. Dataset Creation

Figure 1 represents the steps involved in creating the dataset.

Firstly we pick a base pre-trained LLM, which should has some basic reasoning capability and domain knowledge but not yet has a good performance on a reasoning task dataset we are going to build next.

Then we find and filter a prompt dataset on math reasoning tasks with verifiable answers that the base model can hardly solve. The purpose of doing so is to ensure these tasks have not been reward hacked by the base model.

Following this, we use this dataset to train this model via GRPO (Shao et al., 2024) with two different reward functions. The first one gives verifiable rewards that has reward hacking probability as low as possible. The other one is a modified reward hacking function in which deliberate false positive reward signals are produced. Specifically, we give the function a fixed probability $p_{reward\_hacking}$ to return full reward even if the answer is not aligned with the gold one. $p_{reward\_hacking}$ is a hyperparameter. We call the second setting the simulated reward hacking process and use the generated answers from the post-trained LLM as positive samples. For the first setting, we call it the non reward-hacking process so that it generates negative samples, which serves as a control group. A detailed algorithm 1 is given for the reward function used in the reward hacking process.

Algorithm 1 A reward hacking function

```
verified_reward = verifier(LLM_output)
if $random() > p_{reward\_hacking}$ then
    return 1.0
else
    return verified_reward
end if
```

3.2. Multi-vector embedding based classification

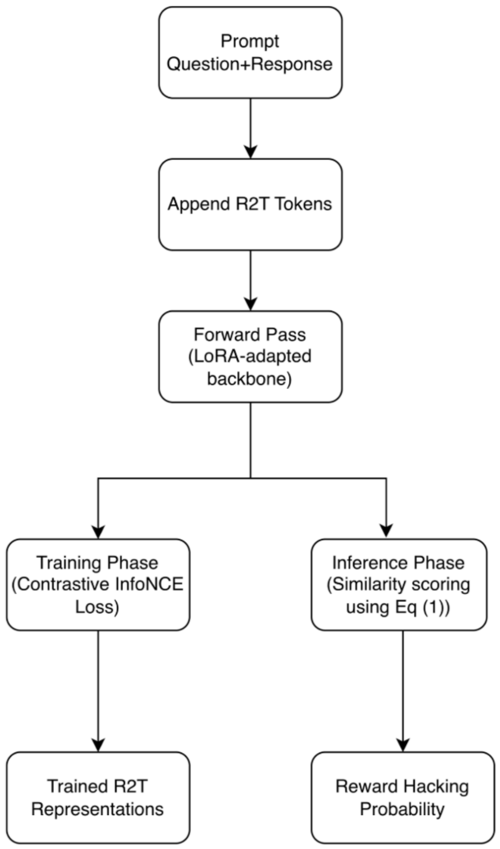

Prompt Question+Response
Append R2T Tokens
Forward Pass (LoRA-adapted backbone)
Training Phase (Contrastive InfoNCE Loss)
Inference Phase (Similarity scoring using Eq (1))
Trained R2T Representations
Reward Hacking Probability

Figure 2. Training and Inference process of R2T

This work is inspired by (Park et al., 2025), which is a single vector method for LLM hallucination detection. Based on a similar idea, we try to represent the samples that are reward hacking and those who are not using some embeddings learned from a LLM, by appending some fixed length group of trainable tokens to the end of the LLM input. Figure 2 elucidates the steps involved in the process. The LLM's weight is also open for update so we used LoRA for efficiency. We choose to use the framework from (Xiao et al., 2025), which is a multi-vector based representation learning method which works like the Matryoshka (Kusupati et al., 2022) - an interesting doll set famous in Russia. The scoring function for the multi-vector queries and candidates are defined in Eq 1. This is also used during inference time for predicting reward hacking probabilities on test samples.

$$s^{(g)}(q,c) \;=\; \sum_{i=1}^{r_q^{(g)}} \max_{j\in[1,\,r_c^{(g)}]} \left\langle E_q^{(g,i)},\; E_c^{(g,j)} \right\rangle \tag{1}$$

In Eq 1, q is a query, which is a prompt in this work. c is the generated answers. The reward hacking probability can be calculated by summing up the inner product of query and candidate vectors under different group length. g is the group index, r is the group length and i,j are the vector indexes. For the first k vectors in a group, they are shared with the other groups which have a length no shorter than k.

In a brief explanation, we can think of that the vectors under different group length as Matryoshkas with different sizes. The longer vectors groups may have higher representation capabilities, which look like the larger dolls. But these groups actually share some of the vectors with the shorter length groups, as the larger dolls can somehow cover the "smaller ones" by sharing a similar shape.

We also give the contrastive learning based loss function from the framework, which is given in Eq 2.

$$\mathcal{L}_{\text{NCE}}^{(g)} = -\frac{1}{B}\sum_{u=1}^{B} \log \frac{\exp\left(s^{(g)}(u,u)\right)}{\exp\left(s^{(g)}(u,u)\right)+\sum_{v\neq u}\exp\left(s^{(g)}(u,v)\right)+\exp\left(s^{(g)}(q^{(u)},c^{(u,-)})\right)} \tag{2}$$

Here u and v are different data indices within a batch, while $(u,-)$ is the negative sample index for prompt u. By summing up the loss of different groups, we have a final loss function as Eq 3, where ω_g is the weight for group g.

$$\mathcal{L}_{final} = \sum_g \omega_g \mathcal{L}_{\text{NCE}}^{(g)} \tag{3}$$

4. Experiments

4.1. Implementation

We firstly build a filtered math dataset based on (Albalak et al., 2025). Specifically, only the difficult problems for the base model are chosen. Then we run RL experiments on it and build a dataset with a few hundred of prompts with reward hacking answers and non reward hacking answers. We set $p_{reward\_hacking}$ = 0.5 in this work.

We used the Qwen2.5-Instruct (Qwen et al., 2025) series of models for RL training, and trained various versions of R2T models under different backbones, including Qwen3 (Yang et al., 2025) and GPT-OSS (OpenAI, 2025) for comparison.

4.2. Training

The training curve with the Qwen3-8B backbone is presented in Fig 3. We can see that the loss for the longer groups converges faster than the shorter ones, demonstrating the strengths of multi-vector. We finetuned the training script of R2T so that it can be completed in a single GPU, which can be found in the source code. We found that there are some instability issues during the training, which we leave to address in the future.

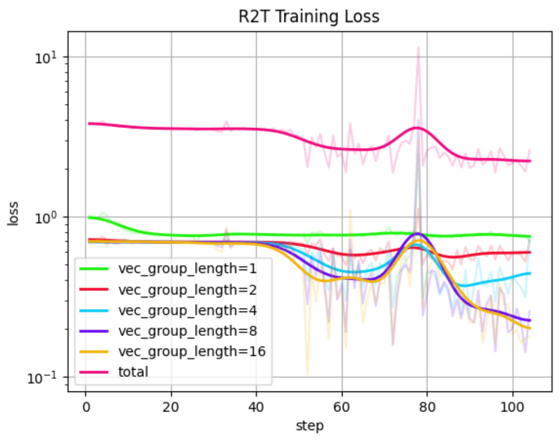
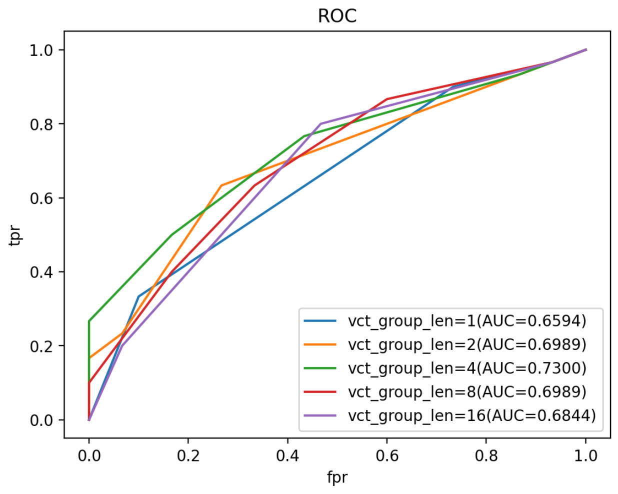


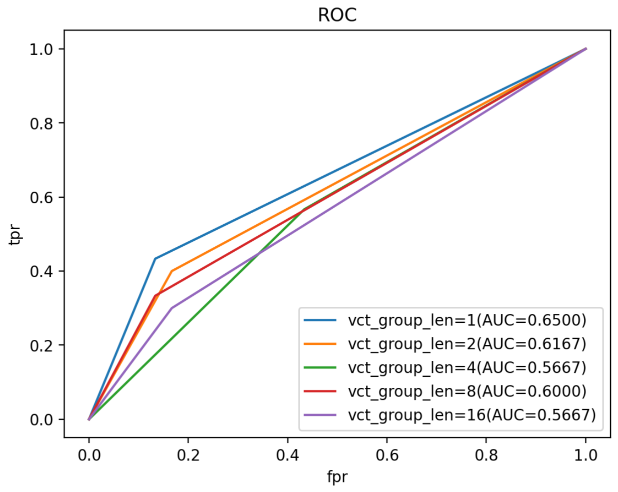
Figure 3. The training loss of R2T with Qwen3-8B under different vector group length.

4.3. Testing

We evaluated the performance of the R2T models for a test dataset in Fig 4. The best AUC is 0.73 achieved by Qwen3-8B backbone under the vector group of length 4. We can see that from our experiments extending the group length further beyond 4 does not necessarily give a better result.



(a) Qwen3-8B



(b) GPT-OSS-20B

Figure 4. Testing ROC and AUC on different backbones.

4.4. Scaling

We trained and tested R2T performance under Qwen3 backbones of different scales in Fig 5. We can see that the performance of R2T correlates well with model parameter size. In Fig 4b, we gives the result of R2T using a GPT-OSS-20B backbone under 4-bit quantization and with QLoRA(Dettmers et al., 2023). While all other training settings remain the same, two series of backbone LLMs present different AUC performances.

5. Conclusion

In this paper, we present a general framework for detecting reward hacking in LLM reasoning tasks. Firstly, we construct a data synthesis pipeline that generates both reward-hacking and non-reward-hacking samples. Subsequently, we employ a multi-vector retrieval–based approach to train R2T, a fixed-length set of soft tokens, for the classification of these synthesized samples. Experiments across multiple backbone models demonstrate the feasibility and effectiveness of the proposed framework.

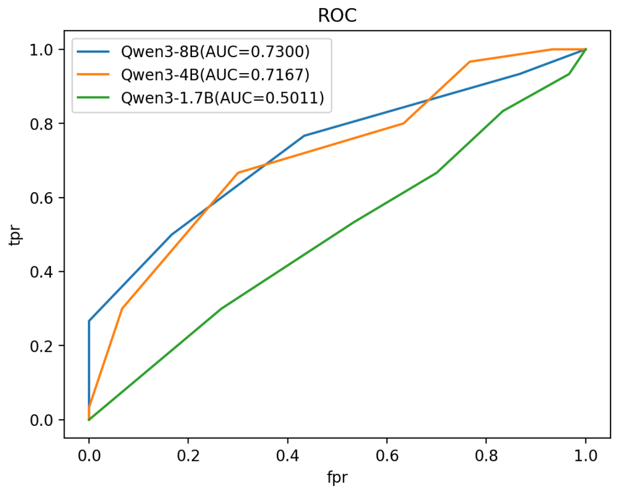


Figure 5. R2T of different model scales

Despite these promising results, several limitations remain. First, the scale of the synthesized dataset is constrained by computational budget limitations. Second, the training dynamics of both the reinforcement learning component and the soft-token optimization process exhibit instability, although convergence is ultimately achieved. Thirdly, a longer R2T group length or a larger backbone model parameter size do not necessarily guarantee a better performance in our experiments. These issues are possible directions for future exploration.

References

Albalak, A., Phung, D., Lile, N., Rafailov, R., Gandhi, K., Castricato, L., Singh, A., Blagden, C., Xiang, V., Mahan, D., and Haber, N. Big-math: A large-scale, high-quality math dataset for reinforcement learning in language models, 2025. URL https://arxiv.org/abs/2502.17387.

Baker, B., Huizinga, J., Gao, L., Dou, Z., Guan, M. Y., Madry, A., Zaremba, W., Pachocki, J., and Farhi, D. Monitoring reasoning models for misbehavior and the risks of promoting obfuscation, 2025. arXiv preprint.

Bukharin, A., Qian, H., Sun, S., Renduchintala, A., Singhal, S., Wang, Z., Kuchaiev, O., Delalleau, O., and Zhao,

T. Adversarial training of reward models, 2025. arXiv preprint.

Dai, C., Li, K., Zhou, W., Hu, S., Hu, T., Zhu, W., and Yan, Y. Reward hacking in reinforcement learning and rlhf: A multidisciplinary examination of vulnerabilities, mitigation strategies, and alignment challenges, 2025. arXiv preprint.

Dettmers, T., Pagnoni, A., Holtzman, A., and Zettlemoyer, L. Qlora: Efficient finetuning of quantized llms, 2023. URL `https://arxiv.org/abs/2305.14314`.

Fan, L., Zhang, Y., Chen, M., and Liu, Z. Posterior-grpo: Rewarding reasoning processes in code generation, 2025. arXiv preprint.

Fu, J., Zhao, X., Yao, C., Wang, H., Han, Q., and Xiao, Y. Reward shaping to mitigate reward hacking in rlhf, 2025. arXiv preprint.

Kiela, D., Wang, C., and Cho, K. Dynamic meta-embeddings for improved sentence representations. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, 2018.

Kusupati, A., Bhatt, G., Rege, A., Wallingford, M., Sinha, A., Ramanujan, V., Howard-Snyder, W., Chen, K., Kakade, S., Jain, P., and Farhadi, A. Matryoshka representation learning, 2022. arXiv preprint.

Miao, Y., Zhang, S., Ding, L., Bao, R., Zhang, L., and Tao, D. Inform: Information-theoretic reward modeling for mitigating reward hacking, 2024. arXiv preprint.

OpenAI. gpt-oss-120b & gpt-oss-20b model card, 2025. URL `https://arxiv.org/abs/2508.10925`.

Park, S., Du, X., Yeh, M.-H., Wang, H., and Li, Y. Steer llm latents for hallucination detection, 2025. arXiv preprint.

Qwen, :, Yang, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Li, C., Liu, D., Huang, F., Wei, H., Lin, H., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Lin, J., Dang, K., Lu, K., Bao, K., Yang, K., Yu, L., Li, M., Xue, M., Zhang, P., Zhu, Q., Men, R., Lin, R., Li, T., Tang, T., Xia, T., Ren, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Wan, Y., Liu, Y., Cui, Z., Zhang, Z., and Qiu, Z. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

Ruan, C., Jiang, D., Wang, Y., and Chen, W. Critique-coder: Enhancing coder models by critique reinforcement learning, 2025. arXiv preprint.

Shao, Z., Wang, P., Zhu, Q., Xu, R., Song, J., Bi, X., Zhang, H., Zhang, M., Li, Y. K., Wu, Y., and Guo, D. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024. URL `https://arxiv.org/abs/2402.03300`.

Turpin, M., Arditi, A., Li, M., Benton, J., and Michael, J. Teaching models to verbalize reward hacking in chain-of-thought reasoning, 2025. arXiv preprint.

Wang, X., Joshi, N., Plank, B., Angell, R., and He, H. Is it thinking or cheating? detecting implicit reward hacking by measuring reasoning effort. *arXiv preprint arXiv:2510.01367*, 2025a.

Wang, Y., Yue, X., and Chen, W. Critique fine-tuning: Learning to critique is more effective than learning to imitate, 2025b. arXiv preprint.

Weller, O., Boratko, M., Naim, I., and Lee, J. On the theoretical limitations of embedding-based retrieval, 2025. arXiv preprint.

Williams, M., Carroll, M., Narang, A., Weisser, C., Murphy, B., and Dragan, A. On targeted manipulation and deception when optimizing llms for user feedback. *arXiv preprint arXiv:2411.02306*, 2024.

Xiao, Z., Ma, Q., Gu, M., Chen, C.-C., Chen, X., Ordonez, V., and Mohan, V. Metaembed: Scaling multimodal retrieval at test time with flexible late interaction, 2025. arXiv preprint.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., Zheng, B., Yu, B., Gao, C., Huang, C., Lv, C., Zheng, C., Liu, D., Zhou, F., Huang, F., Hu, F., Ge, H., Wei, H., Lin, H., Tang, J., Yang, J., Tu, J., Zhang, J., Yang, J., Yang, J., Zhou, J., Zhou, J., Lin, J., Dang, K., Bao, K., Yang, K., Yu, L., Deng, L., Li, M., Xue, M., Li, M., Zhang, P., Wang, P., Zhu, Q., Men, R., Gao, R., Liu, S., Luo, S., Li, T., Tang, T., Yin, W., Ren, X., Wang, X., Zhang, X., Ren, X., Fan, Y., Su, Y., Zhang, Y., Zhang, Y., Wan, Y., Liu, Y., Wang, Z., Cui, Z., Zhang, Z., Zhou, Z., and Qiu, Z. Qwen3 technical report, 2025. URL `https://arxiv.org/abs/2505.09388`.

Yeo, E., Tong, Y., Niu, M., Neubig, G., and Yue, X. Demystifying long chain-of-thought reasoning in llms, 2025. arXiv preprint.

A. Additional information

A.1. Member Contributions:

Equal Contribution from each member:

1. Joseph - 33.4%
2. Deepa - 33.3%
3. Wei-Shiang - 33.3%

A.2. Course Evaluation Completion

1. Joseph - Yes
2. Deepa - Yes
3. Wei-Shiang - Yes