Exploring Domain Robust Lightweight Reward Models based on Router Mechanism

Hyuk Namgoong[1], **Jeesu Jung**[1], **Sangkeun Jung**[1*] and **Yoonhyung Roh**[2]
[1]Computer Science and Engineering, Chungnam National University, Republic of Korea
[2]Electronics and Telecommunications Research Institute, Republic of Korea
{hyuk199, jisu.jung5, hugmanskj}@gmail.com and yhroh@etri.re.kr

Abstract

Recent advancements in large language models have heavily relied on the large reward model from reinforcement learning from human feedback for fine-tuning. However, the use of a single reward model across various domains may not always be optimal, often requiring retraining from scratch when new domain data is introduced. To address these challenges, we explore the utilization of small language models operating in a domain-specific manner based on router mechanisms. Our three approaches are: 1) utilize mixture of experts to form a single reward model by modularizing an internal router and experts, 2) employing external router to select the appropriate reward model from multiple domain-specific models, and 3) the framework reduces parameter size by loading reward models and router adapters onto a single small language model using adapters. Experimental validation underscores the effectiveness of our approach, demonstrating performance comparable to baseline methods while also reducing the total parameter size.

1 Introduction

Most widely adopted Large Language Models (LLMs) have used the reward model of Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022) for fine-tuning. These reward models are trained from various human feedback domains and are subsequently utilized as evaluation metrics during LLM fine-tuning processes.

However, training a single reward model across various domains to serve multiple purposes may lead to situations where the model is not fit for specific domains. Additionally, there is a challenge of retraining the reward model from scratch when new dataset from a new domain is introduced.

In this paper, we explore various router methods to address these challenges, as summarized in Table 1. Our approach, **M**ixture of **R**eward **E**xperts (MoRE), involves modularizing an internal router and experts within small language models to form a single reward model. **R**outer for **DO**main-specific reward model**S** (RODOS) employs an external router to select the appropriate reward model from multiple domain-specific reward models. The **A**dapter **R**outer **L**ightweight **I**ntegrated reward**S** **S**witching (ARLISS) framework applies adapters to load reward models and router adapters onto a single small language model, thereby reducing the parameter size of the multi-models.

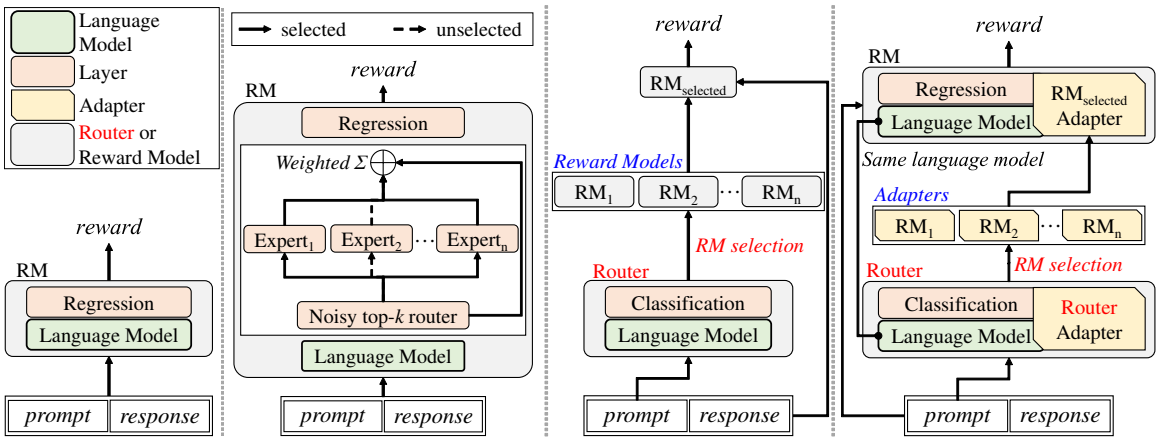
Method	Router	Reward model type	Training	
			Type	Parameter
Baseline	×	Single	Full	All
Base$_{LoRA}$	×	Single	PEFT (LoRA)	Partial
MoRE	○ (Internal)	Single	Full	All
RODOS	○ (External)	Multiple	Full	All
ARLISS	○ (External)	Multiple	PEFT (LoRA)	Partial

Table 1: Comparison of each method for the reward model. Baseline consists of a single reward model without a router. Base$_{LoRA}$ is similar to the baseline but applies Parameter-Efficient Fine-Tuning (PEFT) during training. Mixture of Reward Experts (MoRE) features an internal router but remains a single reward model. Router for DOmain-spcific reward modelS (RODOS) combines multiple reward models with a external router structure. Adapter Router Lightweight Integrated rewardS Switching (ARLISS) framework drastically reduces parameter size by applying PEFT to multiple reward models and external router.

To validate our methodologies, we conducted experiments with five different domains of reward datasets. In this experiment, our methods generally outperform the baseline, while RODOS shows the best performance. MoRE showcases a size reduction of about 52%, while ARLISS achieves a reduction of approximately 55% compared to the baseline.

*Corresponding author

Findings of the Association for Computational Linguistics: ACL 2024, pages 8644–8652



(a) Baseline (b) MoRE (c) RODOS (d) ARLISS

Figure 1: Illustration of each method. RM represents the reward model, and reward is scalar. n denotes the number of domains, and Mixture of Reward Experts (MoRE) refers to Sparse Mixture of Experts with k equal to 2. Router for DOmain-specific reward modelS (RODOS) involves loading all models for use, while the Adapter Router Lightweight Integrated rewardS Switching (ARLISS) framework loads only router and reward model adapters and a single language model, using adapter switching within the same language model.

2 Related Works

Recent research focuses on improving LLMs (Chowdhery et al., 2022; Biderman et al., 2023; Touvron et al., 2023) training efficiency. Introducing the reward model serves to evaluate LLM performance in the RLHF fine-tuning method. In (Ouyang et al., 2022), the reward model spans various domains, while (Black et al., 2023) applies the RLHF method to image generation.

Research has explored methods for routing language models, such as routing LLMs (Shnitzer et al., 2023; Liu and Liu, 2021; Ravaut et al., 2022; Jiang et al., 2023). Furthermore, various studies are underway to modularize and utilize routers within models(Jiang et al., 2024; Dikkala et al., 2023; Peng et al., 2023), with Mixture of Experts (MoE) (Chen et al., 2022) being one.

Research on efficient fine-tuning of language models is ongoing. Low-Rank Adaptation (LoRA) (Hu et al., 2021) attaches adapters to each layer and updates only the adapter parameters, enabling efficient learning. Building upon LoRA, further research explores efficiency improvements(Dettmers et al., 2023; Rajabzadeh et al.; Babakniya et al., 2023) and additional tasks(Zhang et al., 2023; Everaert et al., 2023; Blattmann et al., 2023).

We do not train a single reward model across diverse domains. Instead, we utilize adapters to construct multi-reward models and routers, employing a small language model with LoRA, thereby reducing training time and parameters.

3 Router Based Switching Reward Models

The reward model assigns rewards to prompt and response. In RLHF, the reward model's loss function calculates the difference between the rewards for the chosen and rejected responses. Reward model dataset has the structure of one input prompt and least two of responses.

These reward models cover diverse domains like human preferences and toxic responses, using large-scale models. However, relying solely on one large model may not suit specific domains, and training from scratch for new domains takes time.

3.1 Mixture of Reward Experts

MoRE operates by having an internal router select suitable experts among several options, with both the router and experts modularized internally within the model. To implement MoRE, we utilize sparse MoE(Shazeer et al., 2017), applying to small language models to create a single reward model. Maintaining the structure of a single reward model, it processes all dataset together during training, ensuring a training process similar to traditional method.

Sparse MoE, as depicted in Figure 1b, utilizes noisy top-k gating within the router layer directs the output to multiple expert layers before reaching the output layer. These expert layers follow a feed-forward network structure, computing a weighted sum based on the top-k expert outputs, and then the

regression layer generates the reward. Additionally, layer normalization is applied before the sparse MoE and regression layer.

3.2 Router for Domain-Specific Reward Models

We introduce RODOS, in Figure 1c, which involves training a small language model for each domain to create multiple domain-specific reward models. The external router is trained to select the reward model suitable for each prompt's domain. This resolves the challenge of a single large reward model trained across multiple domains, which may not be suitable for specific domains.

Furthermore, RODOS offers a time-efficient solution by training new reward models for new data and retraining the router, rather than restarting the entire reward model training process. This efficiency is attributed to smaller model sizes and shorter router training times relative to reward model training.

3.3 Adapter Router Lightweight Integrated Rewards Switching Framework

Deploying all reward models and router creates deployment challenges for GPU memory. Hosting various models simultaneously results in the total parameter count becoming a multiple of the model parameters, thus demanding a considerable amount of GPU memory.

In the ARLISS framework, in Figure 1d, all reward models and routers are trained using adapters, with only the adapter parameters retained, and adapters are dynamically switched during inference. The router adapter selects and switches to the appropriate reward adapter during utilization. This approach consolidates multiple reward models and router into a single language model with multiple adapters, thereby reducing the total size of model parameters, making them lightweight.

We utilize Parameter-Efficient Fine-Tuning (Mangrulkar et al., 2022) alongside LoRA, functioning as the adapter mechanism. This enables efficient fine-tuning by updating only adapter parameters, contributing to the overall efficiency of the ARLISS framework.

4 Experiments Setup

4.1 Datasets

In this study, we validate the methodology using reward model datasets from five different domains. In cases where the dataset structure is unsuitable for training a reward model, we convert it to a suitable reward dataset structure using only English data.

Anthropic dataset detects toxic responses and distinguishes whether a response is helpful or harmless (Bai et al., 2022). SHP is a dataset that has two human-written summary responses in a given context (Ethayarajh et al., 2022). HellaSwag is a dataset used for sentence completion tasks, featuring multiple responses to a given prompt (Zellers et al., 2019). Dahoas is a dataset where the model generates two responses to a prompt and humans distinguish between good and bad responses (Alex Havrilla, 2023). Oasst is a dataset that has ranked human-written responses in a given prompt [1].The conversion of each dataset into a reward dataset structure is detailed in Appendix B

4.2 Language Models

We employed the encoder-only model DeBERTaV3(DeB) (He et al., 2021), which leverages Transformer's encoder. For our methods, we implement language models such as DeB$_{base}$, DeB$_{small}$, and DeB$_{xsmall}$. The router model is implement with the same language model as the reward model.

4.3 Baseline Methods

In Table 1, the baseline method is a traditional single reward model trained without a router. This method is implemented using DeB$_{large}$ and DeB$_{base}$ for comparison with our proposed approaches. During fine-tuning, all datasets are processed together. Preliminary experiments with other models are detailed in the Appendix E.

Additionally, Base$_{LoRA}$ was included in the experiments. This method follows the same training process as the baseline but incorporates LoRA. The purpose is to determine if applying LoRA yields higher performance than the baseline DeB$_{large}$. However, it was observed that Base$_{LoRA}$ exhibited lower performance. Base$_{LoRA}$ were conducted using DeB$base$.

4.4 Evaluation Metric for Reward Model

To evaluate the performance of reward model, we utilized $binary\ accuracy$. During reward computation for each prompt-response pair, if the reward for the chosen response exceeds that of the rejected response, it is classified as $true$; otherwise, it is classified as $false$.

[1] https://huggingface.co/datasets/OpenAssistant/oasst2

Method	Language model	Total Parameter (M)	Accuracy					
			Anthropic	SHP	HellaSwag	Dahoas	Oasst	Average
Baseline	DeB_{large}	435	**0.6359** .0058	0.6350 .0117	0.4992 .0009	0.9984 .0003	0.7174 .0053	0.6972 .0048
	DeB_{base}	185 (42.5%)	0.6204 .0031	0.6229 .0054	0.5019 .0025	0.9978 .0008	0.7311 .0060	0.6948 .0036
Base_{LoRA}	DeB_{base}	187 (43.0%)	0.6146 .0053	0.6236 .0083	0.4978 .0012	0.9974 .0007	0.7234 .0072	0.6914 .0030
MoRE	DeB_{base}	207 (47.6%)	0.6205 .0032	0.6265 .0080	0.4995 .0010	0.9972 .0009	**0.7368** .0099	0.6961 .0029
	DeB_{small}	164 (37.7%)	0.6097 .0021	0.6187 .0044	0.4944 .0026	0.9965 .0007	0.7180 .0089	0.6875 .0024
	DeB_{xsmall}	77 (17.7%)	0.5892 .0041	0.6117 .0020	0.5019 .0027	0.9945 .0007	0.7207 .0023	0.6836 .0015
RODOS	DeB_{base}	1,110 (255.2%)	0.6332 .0005	0.6424 .0017	0.4975 .0027	**0.9987** .0000	0.7299 .0002	**0.7003** .0007
	DeB_{small}	846 (194.5%)	0.6236 .0004	0.6367 .0026	0.4969 .0027	0.9981 .0000	0.7290 .0002	0.6969 .0004
	DeB_{xsmall}	420 (96.6%)	0.5927 .0002	0.6301 .0023	**0.5072** .0027	0.9965 .0000	0.6961 .0003	0.6845 .0005
ARLISS	DeB_{base}	197 (45.3%)	0.6254 .0004	**0.6525** .0017	0.4967 .0000	0.9977 .0003	0.7150 .0031	0.6975 .0009
	DeB_{small}	147 (33.8%)	0.6167 .0004	0.6297 .0010	0.4991 .0001	0.9984 .0005	0.7240 .0023	0.6936 .0007
	DeB_{xsmall}	76 (17.5%)	0.6042 .0004	0.6430 .0032	0.5018 .0001	0.9975 .0001	0.7168 .0050	0.6927 .0007

Table 2: Average performance across five domains and the total model parameters for each method. Language models are organized by the DeBERTaV3 (DeB) size. Cyan highlight indicates the best performance per our method within each domain, while **Bold** denotes the best performance across all methods. The parentheses in "Total Parameters" represent the percentage relative to the baseline size. Performances are evaluated with five seeds, and $small\ numbers$ denotes standard deviation.

Method	Language model	Total Parameter (M)	1 epoch train time (sec)						
			Total	Anthropic	SHP	HellaSwag	Dahoas	Oasst	Router
Baseline	DeB_{large} (435)	435	17,392	-	-	-	-	-	-
MoRE	DeB_{base} (184)	207 (47.6%)	5,010	-	-	-	-	-	-
RODOS	DeB_{base} (184)	1,110 (255.2%)	7,355	2,768	696	702	528	235	2,426
ARLISS	DeB_{base} (184)	197 (45.3%)	7,067	2,682	663	672	506	216	2,328

Table 3: Training time for 1 epoch and the total model parameters for each method. Language models are DeBERTaV3 (DeB) used in the experiments, with the original parameter size(M) indicated in parentheses. Baseline and Mixture of Reward Experts (MoRE) are single models, so only the total time is presented. The parentheses in "Total Parameters" represent the percentage relative to the baseline.

5 Experimental Results

Our study investigates the effectiveness of the proposed router methods through experimental analyses, focusing on key aspects: evaluating the router's impact on application performance, analyzing training time across methods, and comparing total parameters with and without ARLISS integration.

5.1 Reward Models Performance

We analyze the accuracy of our proposed framework compared to other methods. In this regard, we conduct statistical significance analysis for each test dataset. To ensure meaningful evaluation, we conduct evaluations with 5 different seeds.

Table 2 displays the accuracy for each dataset's test data and the corresponding average. Generally, when the accuracy is less than 0.02, it is considered statistically similar. Excluding the Anthropic dataset, our methods generally outperform the baseline, with RODOS showing the best performance. Moreover, MoRE and ARLISS demonstrate a size reduction of approximately half of the baseline. This suggests that our methods offer the potential to replace the baseline with smaller model sizes.

5.2 Training Time

We analyze the implementation time for models with and without the router. For multi-reward model methods, we assess the training time for each reward model and the router. For single reward model methods, only the training time for the reward model across all datasets is considered.

Table 3 presents the training time for each method per epoch. Overall, our methods show a reduction in time by approximately 63%, with ARLISS demonstrating around a 5% decrease compared to RODOS.

5.3 Total Parameter Size

We analyze our ARLISS framework along with other methods. In this context, we perform parameter size analysis using the same language model.

Table 3 reveals that the ARLISS framework boasts the smallest parameter size. MoRE showcases a size reduction of about 52%, while ARLISS achieves a reduction of approximately 55% compared to the baseline. Although ARLISS employs a

multi-reward model structure, it features 10 million fewer parameters than MoRE and achieves over an 80% reduction compared to RODOS, another multi-reward model framework.

6 Conclusion

In addressing the limitations of a single large reward model, which can be unsuitable for specific domains and requires retraining when new domain data is introduced, we have implemented router methods. MoRE features an internal router alongside a single small reward model, while RODOS incorporates an external router and domain-specific reward models. These methods effectively mitigate challenges related to domain specificity and the need for retraining when new domain data is introduced. Moreover, the ARLISS framework, with adapters for routers and multi-reward models, shows potential for GPU memory optimization by reducing model size.

Further research will focus on optimizing the ARLISS framework. Additionally, we plan to investigate the integration of the ARLISS framework into MoRE.

Limitation

The ARLISS framework requires more inference time compared to RODOS, as discussed in Appendix D. This delay arises from the router selecting the reward model and switching the adapter within the same language model, resulting in time consumption during the switching process.

Acknowledgements

This work was supported by Institute of Information & Communications Technology Planning & Evaluation(IITP) grant funded by the Korea government(MSIT) (2019-0-00004, Development of semi-supervised learning language intelligence technology and Korean tutoring service for foreigners), the National Research Foundation of Korea(NRF) grant funded by the Korea government(MSIT)(No. 2022R1F1A1071047) and research fund of Chungnam National University.

References

Alex Havrilla. 2023. synthetic-instruct-gptj-pairwise (revision cc92d8d).

Sara Babakniya, Ahmed Roushdy Elkordy, Yahya H Ezzeldin, Qingfeng Liu, Kee-Bong Song, Mostafa El-Khamy, and Salman Avestimehr. 2023. Slora: Federated parameter efficient fine-tuning of language models. *arXiv preprint arXiv:2308.06522*.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. 2022. Training a helpful and harmless assistant with reinforcement learning from human feedback.

Prajjwal Bhargava, Aleksandr Drozd, and Anna Rogers. 2021. Generalization in nli: Ways (not) to go beyond simple heuristics.

Stella Biderman, Hailey Schoelkopf, Quentin Anthony, Herbie Bradley, Kyle O'Brien, Eric Hallahan, Mohammad Aflah Khan, Shivanshu Purohit, USVSN Sai Prashanth, Edward Raff, Aviya Skowron, Lintang Sutawika, and Oskar van der Wal. 2023. Pythia: A suite for analyzing large language models across training and scaling.

Kevin Black, Michael Janner, Yilun Du, Ilya Kostrikov, and Sergey Levine. 2023. Training diffusion models with reinforcement learning.

Andreas Blattmann, Tim Dockhorn, Sumith Kulal, Daniel Mendelevitch, Maciej Kilian, Dominik Lorenz, Yam Levi, Zion English, Vikram Voleti, Adam Letts, Varun Jampani, and Robin Rombach. 2023. Stable video diffusion: Scaling latent video diffusion models to large datasets.

Zixiang Chen, Yihe Deng, Yue Wu, Quanquan Gu, and Yuanzhi Li. 2022. Towards understanding mixture of experts in deep learning.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2022. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*.

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

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Nishanth Dikkala, Nikhil Ghosh, Raghu Meka, Rina Panigrahy, Nikhil Vyas, and Xin Wang. 2023. On the benefits of learning to route in mixture-of-experts models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 9376–9396, Singapore. Association for Computational Linguistics.

Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. 2022. Understanding dataset difficulty with $\mathcal{V}$-usable information. In *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR.

Martin Nicolas Everaert, Marco Bocchio, Sami Arpa, Sabine Süsstrunk, and Radhakrishna Achanta. 2023. Diffusion in style. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 2251–2261.

Pengcheng He, Jianfeng Gao, and Weizhu Chen. 2021. Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing.

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

Albert Q. Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, Gianna Lengyel, Guillaume Bour, Guillaume Lample, Lélio Renard Lavaud, Lucile Saulnier, Marie-Anne Lachaux, Pierre Stock, Sandeep Subramanian, Sophia Yang, Szymon Antoniak, Teven Le Scao, Théophile Gervet, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2024. Mixtral of experts.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692.

Yixin Liu and Pengfei Liu. 2021. SimCLS: A simple framework for contrastive learning of abstractive summarization. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, pages 1065–1072, Online. Association for Computational Linguistics.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. `https://github.com/huggingface/peft`.

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.

Bo Peng, Ben Burns, Ziqi Chen, Srinivasan Parthasarathy, and Xia Ning. 2023. Towards efficient and effective adaptation of large language models for sequential recommendation.

Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. 2019. Language models are unsupervised multitask learners.

Hossein Rajabzadeh, Mojtaba Valipour, Marzieh Tahaei, Hyock Ju Kwon, Ali Ghodsi, Boxing Chen, and Mehdi Rezagholizadeh. Qdylora: Quantized dynamic low-rank adaptation for efficient large language model tuning.

Mathieu Ravaut, Shafiq Joty, and Nancy Chen. 2022. SummaReranker: A multi-task mixture-of-experts re-ranking framework for abstractive summarization. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4504–4524, Dublin, Ireland. Association for Computational Linguistics.

Noam Shazeer, Azalia Mirhoseini, Krzysztof Maziarz, Andy Davis, Quoc Le, Geoffrey Hinton, and Jeff Dean. 2017. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer.

Tal Shnitzer, Anthony Ou, Mírian Silva, Kate Soule, Yuekai Sun, Justin Solomon, Neil Thompson, and Mikhail Yurochkin. 2023. Large language model routing with benchmark datasets.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. 2023. Llama: Open and efficient foundation language models.

Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Well-read students learn better: The impact of student initialization on knowledge distillation. *CoRR*, abs/1908.08962.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence? In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*.

Xuan Zhang, Navid Rajabi, Kevin Duh, and Philipp Koehn. 2023. Machine translation with large language models: Prompting, few-shot learning, and fine-tuning with QLoRA. In *Proceedings of the Eighth Conference on Machine Translation*, pages 468–481, Singapore. Association for Computational Linguistics.

A Hyperparameter Settings

In this section, we provide details of the hyperparameter and LoRA settings in our experiments.

Each model is trained with the same hyperparameters to evaluate under identical conditions. Training utilizes a learning rate of 5.0e-6, a batch size of 32, and 3 epochs, with the AdamW optimizer. However, DeB$_{large}$ is trained with batch size of 8 due to memory limitations.

For LoRA, we established the projection layer for $query$, key, and $value$, along with the $dense$ module. We set the rank to 12, alpha to 768, and dropout to 0.1 based on the layers and dimensions of DeB$_{base}$. The experiments were conducted using Nvidia V100 GPUs.

B Conversion to Reward Dataset Structure

In this section, we discuss the process of converting each dataset into the structure of a reward model dataset. First, we introduce the reward model dataset, which consists of one input prompt and at least two responses. Each response is designated as either chosen or rejected, and the reward model learns to assign higher reward to the chosen response compared to the rejected response when given the prompt and response as input. The requirement of "at least two responses" means that responses must be paired as *chosen* and *rejected*; if there are more than two responses, ranking or selecting is performed to pair them into sets.

Anthropic resembles the reward model dataset but combines the prompt and response. To facilitate the training of a reward model, we preprocess it by separating human input as the prompt and the Assistant's response as responses, resulting in a format of one prompt and two responses.

SHP consists of two human-written summary responses in a given context. Based on the desired human-written summary label, we select chosen and rejected responses for the context.

HellaSwag involves sentence completion tasks with more than two endings. We designate the correct endings as chosen and randomly select from the incorrect endings as rejected responses.

Dahoas and Oasst did not require separate conversion into reward model datasets. However, since our experiments were conducted in English, we only used English data from Oasst, which contains multiple languages.

C Size of Datasets

In this section, Table 4 and 5 presents the sizes of the datasets used in the experiments. These datasets are used for train and test the reward models and router.

Dataset	# of data	% of data
Anthropic	80,307	57.02
SHP	19,493	13.84
HellaSwag	19,952	14.17
Dahoas	14,913	10.59
Oasst	6,176	4.39
Total	140,841	100

Table 4: Data size used to train the reward model and router. The number of data for each domain as a percentage of the total training data.

Dataset	# of data	% of data
Anthropic	8,539	34.59
SHP	2,166	8.77
HellaSwag	10,003	40.52
Dahoas	3,313	13.42
Oasst	668	2.71
Total	24,689	100

Table 5: Data size used to test the reward model and router. The number of data for each domain as a percentage of the total testing data.

D Inference Time

In this section, Table 6 provides the inference times for each method and language model. The experiments were conducted using a total of 2500 data samples. We measure the time it takes for the method to process one input from each dataset.

Method	Language model	1step(sec)
Baseline	DeB$_{large}$	0.08
MoRE	DeB$_{base}$	0.04
	DeB$_{small}$	0.02
	DeB$_{xsmall}$	0.04
RODOS	DeB$_{base}$	0.08
	DeB$_{small}$	0.05
	DeB$_{xsmall}$	0.08
ARLISS	DeB$_{base}$	0.19
	DeB$_{small}$	0.10
	DeB$_{xsmall}$	0.19

Table 6: The inference time is measured for each method and language model. We select 500 samples from each of the five test datasets used in the experiment, measure the inference time, and calculate the average.

E Preliminary Model Selection Experiments

In this section, Table 7 presents the results of preliminary experiments conducted to determine the models to be used in subsequent experiments.The Baseline method was applied using DeBERTaV3 and four other models: BERT$_{base}$ (Devlin et al., 2018), BERT$_{small}$ (Bhargava et al., 2021; Turc et al., 2019), RoBERTa$_{base}$ (Liu et al., 2019), and GPT-2 (Radford et al., 2019). These results help in assessing the performance and suitability of each model for the primary experiments. The experiments are conducted with five seeds each, and the performance metrics are averaged and standard deviation is computed accordingly.

Language model	Parameter Size (M)	Accuracy					
		Anthropic	**SHP**	**HellaSwag**	**Dahoas**	**Oasst**	**Average**
DeB$_{large}$	435	0.6359 .0058	0.6350 .0117	0.4992 .0009	0.9984 .0003	0.7174 .0053	0.6972 .0048
DeB$_{base}$	185	0.6204 .0031	0.6229 .0054	0.5019 .0025	0.9978 .0008	0.7311 .0060	0.6948 .0036
DeB$_{small}$	141	0.6046 .0052	0.6213 .0035	0.4926 .0021	0.9963 .0011	0.7156 .0097	0.6861 .0043
DeB$_{xsmall}$	70	0.5853 .0051	0.6165 .0061	0.5016 .0016	0.9956 .0007	0.7213 .0022	0.6841 .0031
BERT$_{base}$	109	0.6157 .0042	0.6095 .0050	0.4993 .0032	0.9951 .0009	0.7087 .0083	0.6857 .0043
BERT$_{small}$	29	0.5857 .0032	0.6156 .0070	0.4986 .0020	0.9917 .0011	0.7117 .0080	0.6807 .0043
RoBERTa$_{base}$	125	0.6241 .0029	0.6194 .0058	0.4973 .0009	0.9974 .0008	0.7162 .0126	0.6909 .0046
GPT-2	124	0.5987 .0031	0.6206 .0064	0.4954 .0018	0.9925 .0021	0.6904 .0124	0.6795 .0052

Table 7: Average performance across five domains and model parameter sizes for experiments using the Baseline method. The language models include DeBERTaV3 (DeB) as used in the paper, BERT$_{base}$, BERT$_{small}$, RoBERTa$_{base}$, and GPT-2. Performances are evaluated with five seeds, and $small\ numbers$ denotes standard deviation.