NLP ODS course Final Project

Sergey Ishmuratov

May 2025

Abstract

This is a final project for ODS NLP course. Source code is avaliable at `https://github.com/SergeyIsh/ods_nlp_final_project/tree/main`.

1 Introduction

LLM Arena is a platform where different large language models (LLMs) compete by generating responses to the same prompts, and human or AI evaluators judge which output is better. This setup helps to compare model performance in an interactive, head-to-head format, often revealing nuanced differences in response quality, coherence, and usefulness.

A reward model is a machine learning model trained to predict human preferences between LLM outputs. By learning from comparison data (e.g., "Response A is better than Response B"), it assigns scores or probabilities indicating which response is more aligned with human judgment. Reward models are crucial for:

- Fine-tuning LLMs (for instance RLHF - Reinforcement Learning from Human Feedback)
- Automating evaluations in benchmarks like LLM Arena
- Improving AI assistants by optimizing for high-quality, preferred outputs

In this paper, I fine-tune DeBERTa-v3 model with LoRA to act as a reward model, predicting the probability that one LLM's response is better than another's based on human preference data. This enables scalable, automated quality assessments for LLM outputs.

1.1 Team

The full project was done by Ishmuratov Sergey.

2 Related Work

The topic of building reward models is flourishing in the NLP articles. The basic idea of using RL from human preferences belongs [Christiano et al., 2017]. [Chang et al., 2024] create a big survey of existing approaches. Also I want to notice that, on the other hand, [Bai et al., 2022] find that human feedback might be harmful for reward model training. Thus, some modern models don't use himan labels to learn reward. For instance, deepseek use mostly synthetic data, as described in [Guo et al., 2025].

The comparisson of quality of existing methods can be found on reward bench leaderboard presented by [Lambert et al., 2024].

3 Model Description

I fine tune a pretrained Deberta v3 for this task. A basic idea of BERT is to use transformer encoder stacks with bidirectional self-attention to learn contextual word representations. The details are described by [Devlin et al., 2019] . I use Deberta v3 ([He et al., 2021]) with ELECTRA-Style Pre-Training. To fine tune the model, I use low-rank adoptation method ([Hu et al., 2022]). I choose query_proj, and value_proj as target for LORA which allow me to train less than 1 percent of params.

Model Structure:

`PeftModelForSequenceClassification` →

- `base_model`: `LoraModel` →
  - `model`: `DebertaV2ForSequenceClassification` →
    - `deberta`: `DebertaV2Model` →
      - `embeddings`:
      - `word_embeddings`: Embedding(128003, 768)
      - `LayerNorm`: LN(768, $\epsilon = 1e^{-7}$)
      - `dropout`: Dropout($p = 0.1$)
      - `encoder`: `DebertaV2Encoder` →
      - `layer`: ModuleList[12 × `DebertaV2Layer`] →
      - `attention`: `DebertaV2Attention` →
      - `self`: `DisentangledSelfAttention` →
      - `query_proj`: `lora.Linear` →
      - `base_layer`: Linear(768 → 768)
      - `lora_A`: Linear(768 → 8, no bias)
      - `lora_B`: Linear(8 → 768, no bias)
      - `lora_dropout`: Dropout($p = 0.1$)
      - `key_proj`: Linear(768 → 768) *(frozen)*
      - `value_proj`: `lora.Linear` →
      - Same structure as `query_proj`
      - `output`: `DebertaV2SelfOutput` →
      - Linear + LayerNorm + Dropout
      - `intermediate`: `DebertaV2Intermediate` →
      - Linear(768 → 3072) + GELU
      - `output`: `DebertaV2Output` →
      - Linear(3072 → 768) + LayerNorm + Dropout
      - `rel_embeddings`: Embedding(512, 768)
      - `LayerNorm`: LN(768, $\epsilon = 1e^{-7}$)
    - `pooler`: `ContextPooler` →
      - Linear(768 → 768) + Dropout
    - `classifier`: `ModulesToSaveWrapper` →
      - Linear(768 → 3) *(trainable)*

Key LoRA Parameters:

- Rank $r = 8$ (for `lora_A`/`lora_B`)
- Dropout $p = 0.1$
- Target Modules: `query_proj`, `value_proj`

Algorithm 1: LoRA-Adapted DeBERTa-v2 Architecture for Sequence Classification

4 Dataset

To train the model I use kaggle dataset `https://www.kaggle.com/competitions/llm-classification-finetuning`. This dataset contains user interactions from the ChatBot Arena. In each user interaction a judge provides one or more prompts to two different large language models, and then indicates which of the models gave the more satisfactory response. The goal of the competition is to predict the preferences of the judges and determine the likelihood that a given prompt/response pair is selected as the winner.

The data contains columns

- id - A unique identifier for the row.
- model_[a/b] - The identity of model_[a/b]. Included in train.csv but not test.csv.
- prompt - The prompt that was given as an input (to both models).
- response_[a/b] - The response from model_[a/b] to the given prompt.
- winner_model_[a/b/tie] - Binary columns marking the judge's selection. The ground truth target column.

5 Experiments

5.1 Metrics

The main metric is cross-entropy loss (log-loss). I also use accuracy and predicted class distribution to track that the model is balanced.

5.2 Experiment Setup

The hyperparameters of my realization:

- learning_rate=2e-5
- per_device_train_batch_size=8
- per_device_eval_batch_size=16
- num_train_epochs=5
- weight_decay=0.01
- warmup_ratio=0.1
- metric_for_best_model="eval_log_loss"
- gradient_accumulation_steps=2
- fp16=True

The model was trained on GPU P100 Kaggle accelerator.

5.3 Baselines

As a baseline I use random forest classifier on a model name and prompt / answer length and word counts. The log loss in 1.21.

6 Results

The log loss of this model was 1.12, which is not the best result in the competition. I struggle with problems of failing by memory notebooks so final model learn only few epochs.

7 Conclusion

I successfully fine-tune DeBERTA v3 as a reward model on kaggle dataset. The quality is not the best because I train only a few epochs. However, using this approach, a real reward model for text generation task can be created.

References

[Bai et al., 2022] Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. (2022). Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

[Chang et al., 2024] Chang, Y., Wang, X., Wang, J., Wu, Y., Yang, L., Zhu, K., Chen, H., Yi, X., Wang, C., Wang, Y., et al. (2024). A survey on evaluation of large language models. *ACM transactions on intelligent systems and technology*, 15(3):1–45.

[Christiano et al., 2017] Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. (2017). Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, pages 4171–4186.

[Guo et al., 2025] Guo, D., Yang, D., Zhang, H., Song, J., Zhang, R., Xu, R., Zhu, Q., Ma, S., Wang, P., Bi, X., et al. (2025). Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning. *arXiv preprint arXiv:2501.12948*.

[He et al., 2021] He, P., Gao, J., and Chen, W. (2021). Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing. *arXiv preprint arXiv:2111.09543*.

[Hu et al., 2022] Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., Chen, W., et al. (2022). Lora: Low-rank adaptation of large language models. *ICLR*, 1(2):3.

[Lambert et al., 2024] Lambert, N., Pyatkin, V., Morrison, J., Miranda, L., Lin, B. Y., Chandu, K., Dziri, N., Kumar, S., Zick, T., Choi, Y., et al. (2024). Rewardbench: Evaluating reward models for language modeling. *arXiv preprint arXiv:2403.13787*.