Prototypical Reward Network for Data Efficient Model Alignment

Anonymous ACL submission

Abstract

The reward model for Reinforcement Learning from Human Feedback (RLHF) has proven effective in fine-tuning Large Language Models (LLMs). This paper explores enhancing RLHF with Prototypical Networks to improve reward models. We propose a framework utilizing Prototypical Networks to enhance reward models under limited human feedback, enabling more stable and reliable structural learning from fewer samples. This enhances the model's adaptability and accuracy in interpreting human preferences. Our experiments demonstrate that this approach significantly improves the performance of reward models and LLMs in human feedback tasks, surpassing traditional methods, especially in data-limited scenarios.

1 Introduction

Reinforcement Learning from Human Feedback (RLHF) is a crucial technique that combines human intuitive judgment with the model's capacity for large-scale data processing (Cortes et al., 2015; Bai et al., 2022a; Stiennon et al., 2020). This approach allows language models to better understand and adapt to human communication styles and preferences (Yuan et al., 2023). By utilizing Reinforcement Learning (RL) instead of supervised fine-tuning, RLHF captures the complexity of human language, which involves emotions, context, and subtle linguistic differences (Ouyang et al., 2022). This results in greater adaptability and flexibility in interactions with humans.

In Reinforcement Learning from Human Feedback, the learning of the reward model is crucial and typically requires a substantial amount of data for effective training (Wang et al., 2024; Lee et al., 2023; Bai et al., 2022b; Gilardi et al., 2023). A high-quality reward model is essential to ensure the accuracy and efficiency of the RLHF learning process (Ouyang et al., 2022). Particularly in complex Reinforcement Learning environments, a well-tuned reward model can guide the model to learn along the correct path, preventing deviation from the target (Paulus et al., 2017). However, if the quality of the reward model is inadequate, it may learn a complex and inaccurate surface, leading the model to discover high-scoring yet inaccurate points during the Reinforcement Learning process (Chen et al., 2019; Li, 2017). This could result in the model "overfitting" the reward model by generating peculiar outputs to maximize rewards. In such cases, we may end up with a strange strategy that, although scoring high, is misleading and deviates from the actual objectives (Wang et al., 2021). This can significantly cause the RLHF learning outcomes to stray far from human preferences.

To solve these challenges, we propose the Prototypical Reward Model (Proto-RM). Prototypical Networks are instance-based learning algorithms that learn representative prototypes for each class to perform classifications or other tasks (Snell et al., 2017). These networks are particularly suitable for few-shot learning scenarios, as they efficiently extract key features from limited samples and use them for decision-making (Liu et al., 2020). By optimizing the embedding process in the reward model using Prototypical Networks, we leverage the strengths of Prototypical Networks in few-shot learning. This integration enables the reward model to learn more stable and reliable data representation structures with limited sample sizes. Particularly in enhancing the model's learning and generalization from human feedback samples, this method is especially suitable, given the limitations of sample quantity and the complexity of human preferences (Bai et al., 2022b).

To enhance the effectiveness of the reward model within limited human feedback data, we explore a range of methods. These methods aim to decrease reliance on human feedback without diminishing the performance of the reward model. The funda-

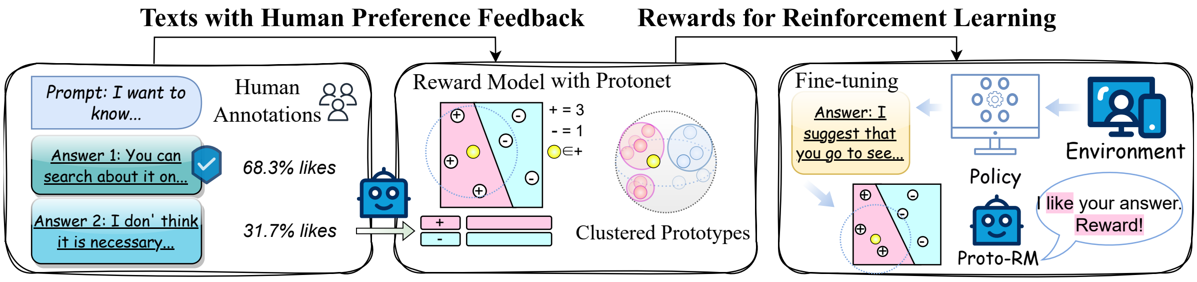

Figure 1: Framework of 1) enhanced reward model with 2) Prototypical Network 3) fine-tuning language models.

mental principle of the reward model is to learn from human feedback to evaluate and guide the output of the model, ensuring it aligns with human expectations and standards. Its key capability lies in effectively learning and extracting vital parameter information from limited human feedback, thus guiding the model's behavior. Therefore, we need to preserve and maximize the use of the network structure and parameterization capabilities of the reward model.

In this context, we need a method that performs well in small-sample learning scenarios, which is suitable for learning from human feedback samples and does not affect the network structure of the reward model.

Our method can be summarized in three key steps: First, we do **Sample Encoding and Prototype Initialization.** We encode samples using the reward model. We first initialize a set of prototypes using a subset of sample encodings. Then, we compare and relate the encodings of other samples with these initialized prototypes. Second, we go through **Prototype Update and Addition.** The sample encoding is updated based on the probability calculated from its distance to the prototypes. We adjust the reward model's parameters by validating the effectiveness of predictions made with updated sample encodings. Continuous updating and refining of prototypes ensure they accurately represent the characteristics of the samples. More effective prototypes lead to better updates of sample encodings, thus enhancing the learning from human feedback samples. Finally, we adopt the **Reward Model Fine-tuning.** With the prototypes and encodings generated in the above process, we train the reward model to more precisely evaluate and guide the output of the language model, thereby improving the performance of LLMs during the fine-tuning process.

Our main contributions are as follows:

1. We propose a structure using the Prototypical Network to improve the reward model. This structure allows for training with fewer human feedback samples without compromising the learning ability of the reward model in scenarios with ample samples.

2. We explore a prototypical learning method for human feedback samples. This method is effective in handling human feedback that is difficult to quantify and varies in length.

3. We conduct a series of experiments to validate the effectiveness and robustness of our method (Proto-RM) across different dataset sizes and evaluate the performance of LLM fine-tuned by Proto-RM. The experiments demonstrate that our method exhibits significant advantages and achieves the effectiveness of using more samples, even with limited samples.

2 Related Work

2.1 Reinforcement Learning from Human Feedback

RLHF is a vital component in training advanced Large Language Models (LLMs) (Christiano et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022; Casper et al., 2023), such as OpenAI's GPT-4 (Achiam et al., 2023), Google's Bard (Singh et al., 2023), and Meta's Llama 2-Chat (Touvron et al., 2023). RLHF and similar methods enable LLMs to adjust their distributions of texts so that the model outputs are more favored by human evaluators (Song et al., 2023).

RLHF combines three interconnected processes: feedback collection, reward modeling, and policy optimization. After collecting assessments of model outputs from humans, the reward modeling process uses supervised learning to train a reward model that mimics these assessments (Lambert et al., 2023; Dong et al., 2019). The policy optimization process fine-tunes the AI system to produce outputs that receive positive evaluations from the reward model (Zheng et al., 2023). RLHF is effective for being relatively easier to identify "good" behavior compared to other methods for specifying or learning rewards. However, the reliance on large volumes of human feedback data for RLHF fine-tuning poses challenges like high costs (Beeching et al., 2023).

2.2 Prototype and Prototypical Network

Prototypical Learning is a powerful approach for improving model interpretability and accuracy in few-shot classification scenarios (Liu et al., 2020; Kim et al., 2014). Numerous researchers have enhanced prototypical networks for category learning (Pan et al., 2019; Ding et al., 2020; Ji et al., 2020). The advantages of Prototypical Networks lie in their simplicity and intuitiveness, enabling rapid adaptation to new samples and categories without the need for extensive data or complex training processes (Fort, 2017). While these networks are commonly used in classification problems with distinct category labels, their application is notably absent in the domain of non-quantitative semantic understanding and text comparison.

3 Problem Formulation

The primary challenge addressed in this work is to train a reward model with limited human-annotated data. With this reward model we can train a policy that generates high-quality texts as judged by humans.

Input. The input of the reward model consists of a dataset of paired human-annotated texts. We define this dataset as $\mathcal{D} = \{(x_i, y_i^+, y_i^-), (z_i^+, z_i^-)\}_{i=1}^N$. Here, N represents the total number of data pairs. For each text pair, $x \in \mathbf{X}$ is the common post for two corresponding summaries y^+ and $y^- \in \mathbf{Y}$, and z^+ and z^- represent the annotations for y^+ and y^-, respectively. The annotations $z^+, z^- \in \mathbf{Z} = \{\text{chosen}, \text{rejected}\}$.

Output. The outputs consist of 1) $s_{(x_i, y_i^+)}$ and $s_{(x_i, y_i^+)}$, which are the predicting score pair of the input example (x, y^+, y^-); 2) the reward model $f_\phi : \mathbf{X} \times \mathbf{Y} \rightarrow \mathcal{E}$, where $\mathcal{E}$ is an embedding space. Here f_ϕ includes embedding process e_ϕ and aligned linear score output process.

4 Methodology

In this process, our key task is to train a reward model to predict which answer $y \in (y^+, y^-)$ is better as judged by a human, given a prompt x.

4.1 Reward Model with Prototypical Network

Reward Model for RLHF. The role of the reward model is to evaluate the quality of outputs generated by the language model and provide feedback that guides the fine-tuning process to align the model's outputs with human preferences. Given the input dataset $\mathcal{D}$, the reward model for RLHF first converts text pairs into encodings in the embedding space $\mathcal{E}$ with parameter ϕ:

$$\mathbf{e}_\phi(x, y) \rightarrow \mathbf{e} \in \mathcal{E}, \mathbf{e} = (\mathbf{e}_x, \mathbf{e}_y) \quad (1)$$

Here, $\mathbf{e}$ is the representation of the input (x, y), $\mathbf{e}_x$ and $\mathbf{e}_y$ are the representations of the prompt and answer, respectively.

Prototypical Network. In the prototypical network, a set of prototype vectors $\mathbf{p}_k$ is categorized into two groups: $\mathbf{p}^+$ and $\mathbf{p}^-$. The classification of each sample pair's embedding $\mathbf{e}_{(x_i, y_i^*)}, y^* \in \{y^+, y^-\}$ is determined by the proportion of these two classes of prototypes within the adjacent prototypes. The embedding $\mathbf{e}_{(x_i, y_i^*)}$ is updated based on all the prototype vectors in their respective category, with weights assigned according to their importance. The importance of prototype $\mathbf{p}_k$ is computed using the distance metric $d(\cdot, \cdot)$:

$$\mathbb{P}(\mathbf{p}_k | (x_i, y_i^*)) \propto \exp(-d(\mathbf{e}_{(x_i, y_i^*)}, \mathbf{p}_k)) \quad (2)$$

where $d(\cdot, \cdot)$ is usually taken as squared L2 distance. We then update the embedding for each sample according to its class. For a sample embedding related to the $\mathbf{p}^*$ prototype, we update its embedding $\mathbf{e}_{(x_i, y_i^*)}$ using all j $\mathbf{p}^*$ prototypes. We express the formula for updating the embedding as:

$$\mathbf{e}_{(x_i, y_i^*)} = \frac{1}{j} \sum_{k=1}^{j} (\mathbb{P}(\mathbf{p}_k | (x_i, y_i^*)) \cdot \mathbf{p}_k) \quad (3)$$

The updated embedding is then transformed into a score within a linear layer.

4.2 Reward Model with Protonet

Prototype Initialization. During the initialization phase, our goal is to reasonably initialize two classes of prototypes $\mathbf{p}_k \in \{\mathbf{p}^+, \mathbf{p}^-\}$. We randomly select n sample pairs and separate them according to their sample labels z_i. Specifically, we

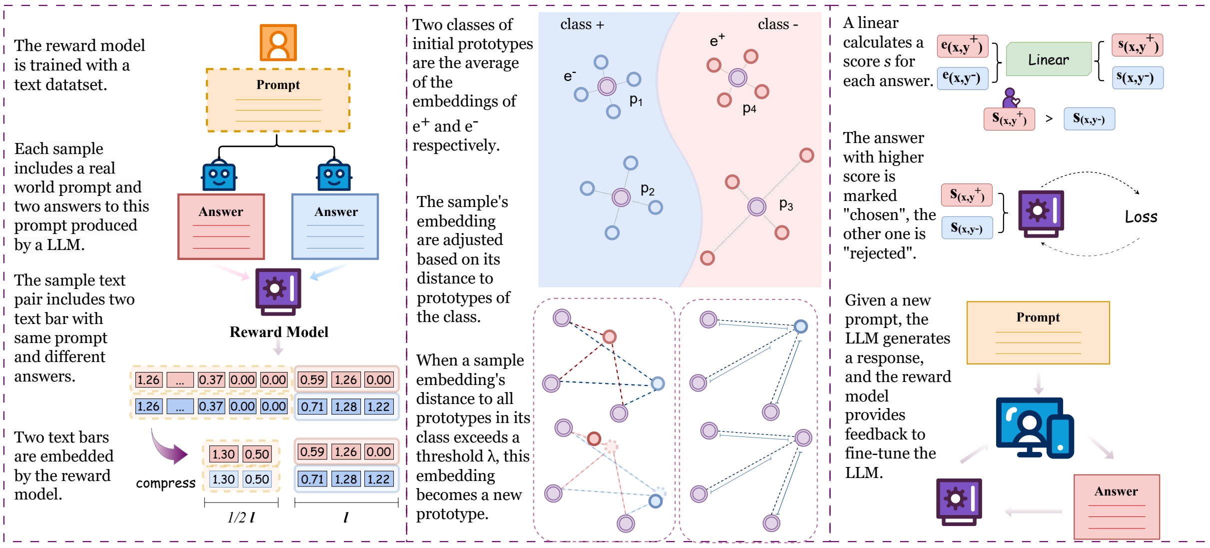

Figure 2: The framework consists of three components: 1) Reward model embedding, 2) Protonet adjustment and 3) RLHF Process. The reward model compress and align the sample text pair embeddings to produce representative prototypes, and the prototypes adjust the embeddings to update the reward model.

initialize different prototypes using the sample embeddings labeled as "chosen" and "rejected". This strategy is employed to allow the model to better learn human preferences as opposed to mere differences in the content of the samples. We process the prompt and answer components of each text bar separately. For those embeddings that initialize $\mathbf{p}_0$, we perform pairwise sample alignment to ensure uniformity and fairness in compression and computation across positive and negative examples. This alignment method guarantees that the prototypes are updated in a consistent manner, reflecting a balanced representation of both prompt and answer components in the embedding space.

$$\mathbf{e}_{x_i} \leftarrow align(\mathbf{e}_{x_i}, \max \|\mathbf{e}_{x_i}\|) \tag{4}$$

where *align* means the embedding $\mathbf{e}_{x_i}$ is updated to a new vector with the same maximum length as the longest embedding vector among all $\mathbf{e}_{x_i}$, and we pad additional elements beyond the original length of $\mathbf{e}_{x_i}$ with zeros. Similarly, we have:

$$\mathbf{e}_{y_i^*} \leftarrow align(\mathbf{e}_{y_i^*}, \max \|\mathbf{e}_{y_i^*}\|) \tag{5}$$

$$\mathbf{e}_{(x_i, y_i^*)} = (\mathbf{e}_{x_i}, \mathbf{e}_{y_i^*}) \tag{6}$$

An initial prototype constructed from n text pairs is defined as $\mathbf{p}_0 = \frac{1}{n}\sum \mathbf{e}_{(x_i, y_i^*)}$, with a length of $\|\mathbf{p}_0\| = (\max \|\mathbf{e}_{x_i}\| + \max \|\mathbf{e}_{y_i^*}\|), i = 1, 2, \ldots n$. This ensures that the prototype encapsulates the essential features of both prompt and answer.

We derive an initial set of K prototypes. In our case, we choose mean pooling as the aggregate function with the parameter of the reward model ϕ frozen. Furthermore, during initialization, we disable the gradient updates of the prototype vectors, ensuring that the initialization is not influenced by other model parameters. This guarantees the robustness of the initialization process.

Prototype Update and Addition. Our goal is to represent the samples effectively and comprehensively by the prototypes. However, a fixed number of prototypes may not suffice for this purpose. Too few prototypes can lead to the loss of important information, while too many prototypes can affect their representativeness and increase computational costs (Snell et al., 2017; Ming et al., 2019).

Therefore, we consider employing the technique of Incremental Mixture Prototypes (IMP) (Allen et al., 2019) to automatically add prototypes, allowing the model to appropriately increase the number of prototypes during training based on the distance relationship between the prototypes and the samples. This technique is commonly used in the classification of graphical samples, but its application in textual information is relatively less frequent. Prototype methods excel in processing graphical samples with their visual and intuitive features, but the abstract and multidimensional characteristics of text, covering semantics, syntax, and context, complicate their use in textual data. Due to our reliable embedding and alignment of text samples using the reward model, we successfully implement IMP for effective learning from human feedback samples.

After initializing the prototypes, we activate the Prototypical Network to better assimilate new input text pairs. To enhance the representativeness and diversity of the prototypes, we 1) appropriately add new prototypes and 2) continually update existing ones.

1) We define the set of prototypes as $\mathbf{P}$. To increase the representativeness and diversity of the prototypes, for each sample $(x_i, y_i^*) \in \mathcal{D}$, if the minimum distance between $\mathbf{e}_{(x_i,y_i^*)}$ and any prototypes in $\mathbf{P}$ exceeds a threshold λ, we create a new prototype based on $\mathbf{e}_{(x_i,y_i^*)}$. The threshold distance λ is defined as:

$$\lambda = 2\sigma \log\left(\frac{\alpha}{(1+\frac{\rho}{\sigma})^{d/2}}\right) \tag{7}$$

where σ is the cluster variance learned jointly with ϕ, ρ is the standard deviation for the base distribution from which the cluster means are sampled, and α is a hyperparameter controlling the concentration of clusters in the Chinese Restaurant Process. Our approach can balance between fitting simple data distributions with low capacity and complex distributions with high capacity.

2) We then compute the Euclidean distance from each text bar in a text pair to every prototype $\mathbf{p}_k$ in their class, denoted as $d(\mathbf{e}_{(x_i,y_i^*)}, \mathbf{p}_k)$. Then, utilizing the negative of these distances, we calculate the softmax to obtain a probability distribution of sample (x_i, y_i^*) belongs to prototype $\mathbf{p}_j$. Additionally, during the update of sample embeddings, we incorporate a proportionate dropout of the prototypes, which enhances the model's ability to generalize and avoid overfitting to specific patterns, as expressed by the following equation:

$$P(\mathbf{p}_i = \mathbf{p}_j | (x_i, y_i^*)) = \frac{\exp(-d(\mathbf{e}_{(x_i,y_i^*)}, \mathbf{p}_j))}{\sum_{k=1}^{\lfloor \rho K \rfloor} \exp(d(\mathbf{e}_{(x_i,y_i^*)}, \mathbf{p}_k))} \tag{8}$$

where ρ is the dropout ratio, and K is the total number of prototypes, $\lfloor \cdot \rfloor$ represents the floor function. Here we compare prototypes within the same class using cosine similarity and drop out the prototypes with the lowest similarity proportionally, instead of random dropout. This method allows the prototypes to be more representative.

After yielding the probabilities with respect to each prototype, we then multiply these probabilities by the embedding of the respective prototype $\mathbf{p}_k$ to obtain the new embedding $\mathbf{e}'_{(x_i,y_i^*)}$ after the Prototypical Network processing.

$$\mathbf{e}'_{(x_i,y_i^*)} = \sum_{k=1}^{K} P(\mathbf{p}_i = \mathbf{p}_k | (x_i, y_i^*)) \cdot \mathbf{p}_k \tag{9}$$

Annotation Prediction. We then evaluate the performance of the model and update it. We predict the annotation z_i of the new embedding $\mathbf{e}'_{(x_i,y_i^*)}$. The embedding transform into a score $s_{(x_i,y_i^*)}$ through a linear layer. By comparing the scores $s_{(x_i,y_i^+)}$ with $s_{(x_i,y_i^-)}$, the model annotate the one with the higher score as "chosen", and the one with the lower score as "rejected". We evaluate the model's predictions z_i against real human annotations and perform backpropagation accordingly.

Loss and Backpropagation. The final step involves the computation of the overall loss, including reward loss and diversity loss to enhance the model's performance and reduce the risk of overfitting.

For reward loss $\mathcal{L}_r$, we adopt the reward loss structure of (Stiennon et al., 2020):

$$\begin{aligned}\mathcal{L}_r = &- \mathbb{E}_{(x_i,y_i^+,y_i^-)\sim\mathcal{Z}}[\log(\sigma(r_\phi(x_i, y_i^+) \\ &- r_\phi(x_i, y_i^-)))]\end{aligned} \tag{10}$$

where $r_\phi(x_i, y_i^*)$ is the scalar output of the reward model for prompt x_i and answer y_i^* with parameter ϕ, and $\mathcal{Z}$ is the collection of human annotations. At the end of training, we normalize the reward model outputs such that the reference text pairs from the dataset achieve a mean score of 0.

For diversity loss $\mathcal{L}_{\text{div}}$, in order to ensure a sparse distribution among prototype points, we employ a hyperparameter τ to constrain the average Euclidean distance between prototype points. As model parameters, prototypes are involved in backpropagation through gradient descent, allowing for dynamic refinement. The sparsity constraint is implemented via a diversity loss $\mathcal{L}_{\text{div}}$ (Ji et al., 2022), which is guided by the average Euclidean distance between prototypes:

$$\psi = \begin{cases} \text{Euc}(\Phi) - \tau & \text{if Euc}(\Phi) \geq \tau, \\ \tau - \text{Euc}(\Phi) & \text{if Euc}(\Phi) < \tau, \end{cases} \tag{11}$$

$$\mathcal{L}_{\text{div}} = \log(\psi + 1). \tag{12}$$

The full objective $\mathcal{L}$ linearly combines $\mathcal{L}_r$ and $\mathcal{L}_{\text{div}}$ using a hyperparameter ρ_d:

$$\mathcal{L} = \mathcal{L}_r + \rho_d \mathcal{L}_{\text{div}} \tag{13}$$

Algorithm 1 Reward model with Protonet

Input: $\mathcal{D} = \{(x_i, y_i^+, y_i^-), (z_i^+, z_i^-)\}_{i=1}^N$, where each $z^+, z^- \in Z = \{\text{chosen}, \text{rejected}\}$
Output: The predicting score pair $S(x, y^+, y^-) = (s^+, s^-)$ and the reward model f_ϕ
Initialize K Prototypes through **Prototype Initialization**
for minibatch $B_r \in \mathcal{D}$ **do**
  Perform **Prototype Update and Addition** and estimate λ according to Eq. 7
  for $(x_i, y_i^+, y_i^-) \in B_r$ **do**
    Converts (x_i, y_i^+, y_i^-) into encodings $\mathbf{e}_{(x_i, y_i^+)}$ and $\mathbf{e}_{(x_i, y_i^-)}$
    for $y_i^* \in y^+, y^-$ **do**
      Allign $\mathbf{e}_{(x_i, y_i^*)}$ according to Eq. 4
      Calculate $d_{i,k} = d(\mathbf{e}_{(x_i, y_i^*)}, \mathbf{p}_k)$ for $\mathbf{p}_k \in \mathbf{p}^*$, and $d_{i,k} = +\infty$ for $\mathbf{p}_k \notin \mathbf{p}^*$
      Update the embedding according to Eq. 3
      if $\min d_{i,k} > \lambda$ **then**
        Create the $K+1$-th prototype $\mathbf{p}_{K+1}$ using $\mathbf{e}_{(x_i, y_i^*)}$; Increment K by 1
      end if
      Compute $s_{(x_i, y_i^*)}$ though **Annotation Prediction**
    end for
  end for
end for

5 Experiments

In this section, we first compare the consistency of annotations between Proto-RM and Baseline Reward Model (Baseline RM) with real human feedback on Prompt-Answer text pairs. Subsequently, we contrast the differences in text quality of LLM outputs after fine-tuning with Proto-RM versus Baseline RM. Following this, we explore the significance of different modules in the learning of the reward model, assessing the effectiveness of our innovative points.

5.1 Experiment Settings

Datasets. We train reward models using three datasets at varying data proportions. The datasets employed are as follows:

	Webgpt	Pairwise	Summarize
5%	979	1,657	9,692
10%	1,958	3,314	19,384
20%	3,916	6,629	38,768
Total	19,578	33,143	193,841

Table 1: Data distribution across different datasets

Webgpt Comparisons (Webgpt) (Nakano et al., 2021) contains pairs of model answers with human preference scores in the WebGPT project.

Synthetic Instruct GPT-J Pairwise (Pairwaise) (Alex et al., 2021) contains human feedback for reward modeling, featuring pairwise summary evaluations and Likert scale quality assessments.

Summarize from Feedback (Summarize) (Stiennon et al., 2020) contains pairwise summaries with human annotations from the TL;DR dataset.

Models. For the pre-trained LLM we adopt the GPT-J model (Wang and Komatsuzaki, 2021). And we use the trlX framework (Havrilla et al., 2023) to implement our algorithm.

Implemetation Settings. In our experiments, we apply a batch size of 8 and initialize each prototype using $n = 2$ examples. The sequence length is set to 550. We fix the value of α at 0.1 and the initial value of ρ at 5. For the optimization process, we use the AdamW optimizer (Zhuang et al., 2022). We search for the best learning rate within the range of $[1e-6, 1e-5]$. Other hyperparameters are set to their default values. All experiments are conducted for a maximum of 5 epochs with early stopping implemented. Regarding hardware, our experiments are run on server equipped with NVIDIA Tesla A100 GPU (80GB memory).

5.2 Comparison with Baseline Reward Model

To compare the performance of Baseline RM and Proto-RM, we train and test both reward models on three datasets by different radios. From Table 2 we can see that, across the different data proportions on the three datasets, Proto-RM consistently surpasses Baseline-RM. On the *Webgpt* dataset, there is an accuracy improvement ranging from 1.48% to 2.15%; on the *Pairwise* dataset, the improvement spans from 0.48% to 0.59%, with Proto-RM nearly achieving perfect accuracy; and on the *Summarize* dataset, especially at the 20% data proportion,

Datasets	Webgpt		Pairwise		Summarize	
RM	Baseline-RM	Proto-RM	Baseline-RM	Proto-RM	Baseline-RM	Proto-RM
5%	57.46 ± 0.21	$\mathbf{58.94 \pm 0.22}$(+1.48)	98.96 ± 0.15	$\mathbf{99.44 \pm 0.18}$(+0.48)	65.36 ± 0.19	$\mathbf{67.67 \pm 0.23}$(+2.31)
10%	58.86 ± 0.24	$\mathbf{59.30 \pm 0.26}$(+0.44)	99.14 ± 0.17	$\mathbf{99.65 \pm 0.20}$(+0.51)	66.51 ± 0.21	$\mathbf{67.76 \pm 0.25}$(+1.25)
20%	58.41 ± 0.28	$\mathbf{60.56 \pm 0.29}$(+2.15)	99.45 ± 0.16	$\mathbf{99.84 \pm 0.11}$(+0.39)	67.46 ± 0.22	$\mathbf{68.72 \pm 0.27}$(+1.26)

Table 2: Comparison with Baseline in different sizes of various datasets, the accuracy of Proto-RM consistently exhibits an exceedance over Baseline-RM.

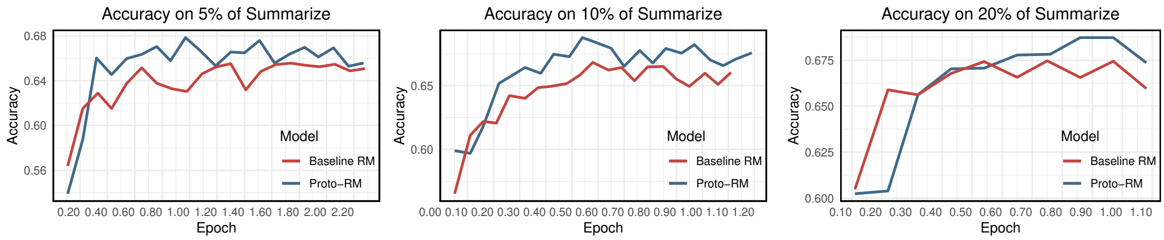

Figure 3: Comparison of reward models' accuracy on 5%, 10%, and 20% datasets.

Proto-RM exhibits the most significant accuracy gain of 1.26%.

The line graphs Figure 4 reinforce the table's data, showcasing that the Proto-RM model maintains a higher accuracy across epochs compared to the Baseline-RM for the 5%, 10%, and 20% of the Summarize dataset. Proto-RM not only starts at a higher accuracy but also demonstrates less variability and ends with a higher accuracy, indicating a more robust model.

5.3 RLHF Performance

To ensure consistency in scoring and to maintain the integrity of the evaluation, all outputs from GPT-J (6B) (Wang and Komatsuzaki, 2021) are assessed by GPT-4 (OpenAI, 2024) across four dimensions, as many studies and attempts to use LLMs for text annotation (Gilardi et al., 2023; Alizadeh et al., 2023; Bai et al., 2022b), indicating that high-quality LLMs are capable of achieving human-like text evaluation abilities (Lee et al., 2023). The scoring standards, which include considerations of factual accuracy, text relevance, information completeness, and clarity of expression, are uniformly applied. Each aspect is scored on a scale up to 10, with increments of 0.5. The overall score is derived as the average of these four individual scores:

Accuracy (Acc): Assesses whether the content of the answer or summary accurately reflects the information and intention of the original prompt.

Relevance (Rel): Checks whether the answer or summary is closely related to the original prompt.

Completeness (Comp): Evaluates whether the provided information is comprehensive, covering all key points and details in the prompt.

Expression (Expr): Considers whether the language expression of the answer or summary is clear and understandable.

The results in Figure 4 indicates that the LLM fine-tuned with Proto-RM outperforms the Baseline across all four aspects, showing an increase from $0.4/10$ to $0.54/10$ in overall score, which is significantly higher than the Baseline. Moreover, it demonstrates a clear advantage in both Accuracy and Expression, with the highest scores reaching $0.76/10$ and $0.82/10$ respectively. Table 3 demonstrates the differences in the output text quality of GPT-J with no fine-tuning, fine-tuned using Baseline-RM, and fine-tuned using Proto-RM. The discrepancies highlighted also validate the efficacy of our improved reward model.

5.4 Ablation Study

Study of IMP. We explore and compare the effects of using different numbers of prototypes with various methods for setting the prototype quantities. Specifically, we examine the outcomes of setting the prototype numbers to twice and thrice the default amount and the outcomes of gradually increasing the number of prototypes from the default to double and triple using the IMP method. Figure 5 illustrates that adopting the IMP method for prototype numbers yields better results in both accuracy and stability compared to fixed prototype numbers. The lines representing IMP methods (both IMP-Double and IMP-Triple) show higher accuracy over the epochs. Additionally, the IMP lines demonstrate a smoother progression with less fluctuation, suggesting greater stability in model performance

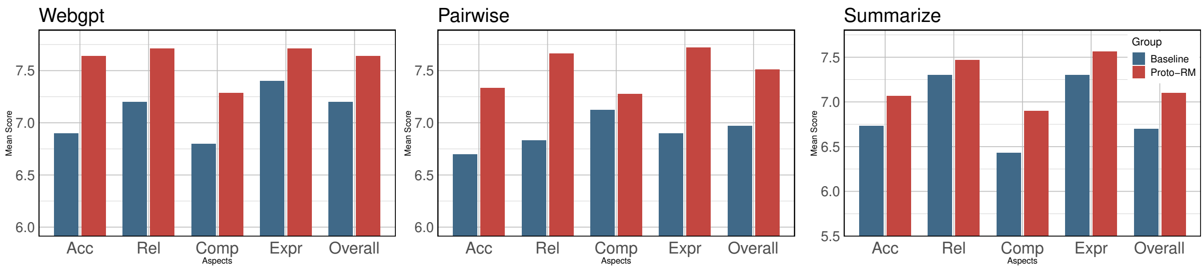

Figure 4: Performance of LLM with reward model fine-tuning.

Prompt: SUBREDDIT: r/relationships TITLE: My girlfriend (20f) of two years cheated on me (20m) by kissing two guys at a Halloween party.
POST: Lately her and I have been having a few problems, and these problems have been brought up before a few times... I feel terrible about it, but this time I was really trying to change for her. For Halloween she went to visit her step brother at a college and I got drunk with my friends and watched movies. Last night (11/1) we got in a huge fight about me not changing and how our relationship won't work out and basically broke up over the phone. So in an effort to try and fix it I drove to her house. She told me how at the parties she went to that two guys kissed her... Should I even try to fix it or would I be better off cutting all ties...
TL;DR:

GPT-J without Fine-tuning	**GPT-J with Baseline Reward Model**	**GPT-J with Protonet-Reward Model**
Girlfriend cheated on me and now we may be fixing things up, but we're not too sure if I should. I love her and we both want to fix things. Should we?	girlfriend of 2 years kissed two guys at Halloween party and I don't know what to do. But I want to try and fix it.	Girlfriend and I broke up over me not changing, decided to fix that, find out she kissed someone else at a Halloween party. Should we work it out or not?

Table 3: Comparative responses from GPT-J models to a given prompt.

across epochs.

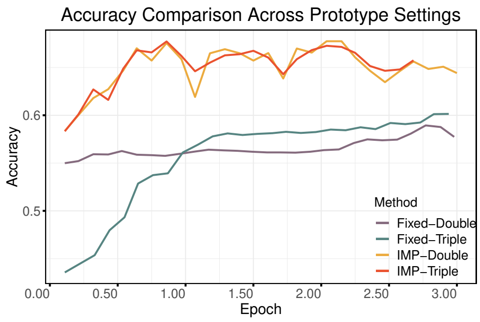

Figure 5: Impact of IMP.

Study of Dropout. Showing in Figure 6, we find that employing a Dropout method, which proportionally drops out a part of the prototypes during the sample embedding updates, yields better results. Specifically, as the line chart illustrates, adopting a Dropout method significantly outperforms the approach of not using Dropout in terms of accuracy. Among the Dropout approaches, the method utilizing Cosine Similarity Dropout achieves higher accuracy compared to Random Dropout and exhibits greater stability. This underscores the effectiveness of using Cosine Similarity Dropout.

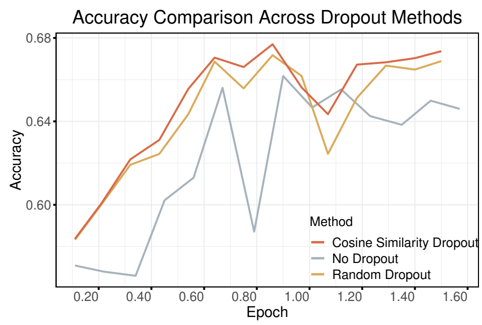

Figure 6: Impact of Dropout.

6 Conclusion

In conclusion, our research demonstrates the efficacy of Prototypical Networks in refining RLHF processes, especially in scenarios with limited human feedback. The enhanced reward model shows a marked improvement in aligning LLM outputs with human preferences, as evidenced by our experimental results. However, our method's application to more diverse and extensive datasets remains an area for future exploration to further validate its effectiveness and adaptability.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Beatrice Alex, Clare Llewellyn, Pawel Orzechowski, and Maria Boutchkova. 2021. The online pivot: Lessons learned from teaching a text and data mining course in lockdown, enhancing online teaching with pair programming and digital badges. In *Proceedings of the Fifth Workshop on Teaching NLP*, pages 138–148, Online. Association for Computational Linguistics.

Meysam Alizadeh, Maël Kubli, Zeynab Samei, Shirin Dehghani, Juan Diego Bermeo, Maria Korobeynikova, and Fabrizio Gilardi. 2023. Open-source large language models outperform crowd workers and approach chatgpt in text-annotation tasks. *arXiv preprint arXiv:2307.02179*.

Kelsey Allen, Evan Shelhamer, Hanul Shin, and Joshua Tenenbaum. 2019. Infinite mixture prototypes for few-shot learning. In *International conference on machine learning*, pages 232–241. PMLR.

Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, et al. 2022a. Training a helpful and harmless assistant with reinforcement learning from human feedback. *arXiv preprint arXiv:2204.05862*.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, et al. 2022b. Constitutional ai: Harmlessness from ai feedback. *arXiv preprint arXiv:2212.08073*.

Edward Beeching, Younes Belkada, Kashif Rasul, Lewis Tunstall, Leandro von Werra, Nazneen Rajani, and Nathan Lambert. 2023. Stackllama: an rl fine-tuned llama model for stack exchange question and answering. *See https://huggingface.co/blog/stackllama (accessed 14 April 2023)*.

Stephen Casper, Xander Davies, Claudia Shi, Thomas Krendl Gilbert, Jérémy Scheurer, Javier Rando, Rachel Freedman, Tomasz Korbak, David Lindner, Pedro Freire, et al. 2023. Open problems and fundamental limitations of reinforcement learning from human feedback. *arXiv preprint arXiv:2307.15217*.

Xinshi Chen, Shuang Li, Hui Li, Shaohua Jiang, Yuan Qi, and Le Song. 2019. Generative adversarial user model for reinforcement learning based recommendation system. In *International Conference on Machine Learning*, pages 1052–1061. PMLR.

Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. 2017. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30.

Corinna Cortes, N Lawarence, D Lee, M Sugiyama, and R Garnett. 2015. Advances in neural information processing systems 28. In *Proceedings of the 29th Annual Conference on Neural Information Processing Systems*.

Kaize Ding, Jianling Wang, Jundong Li, Kai Shu, Chenghao Liu, and Huan Liu. 2020. Graph prototypical networks for few-shot learning on attributed networks. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*, pages 295–304.

Li Dong, Nan Yang, Wenhui Wang, Furu Wei, Xiaodong Liu, Yu Wang, Jianfeng Gao, Ming Zhou, and Hsiao-Wuen Hon. 2019. Unified language model pre-training for natural language understanding and generation. *Advances in neural information processing systems*, 32.

Stanislav Fort. 2017. Gaussian prototypical networks for few-shot learning on omniglot. *arXiv preprint arXiv:1708.02735*.

Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. 2023. Chatgpt outperforms crowd-workers for text-annotation tasks. *arXiv preprint arXiv:2303.15056*.

Alexander Havrilla, Maksym Zhuravinskyi, Duy Phung, Aman Tiwari, Jonathan Tow, Stella Biderman, Quentin Anthony, and Louis Castricato. 2023. trlX: A framework for large scale reinforcement learning from human feedback. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8578–8595, Singapore. Association for Computational Linguistics.

Bin Ji, Shasha Li, Shaoduo Gan, Jie Yu, Jun Ma, and Huijun Liu. 2022. Few-shot named entity recognition with entity-level prototypical network enhanced by dispersedly distributed prototypes. *arXiv preprint arXiv:2208.08023*.

Zhong Ji, Xingliang Chai, Yunlong Yu, Yanwei Pang, and Zhongfei Zhang. 2020. Improved prototypical networks for few-shot learning. *Pattern Recognition Letters*, 140:81–87.

Been Kim, Cynthia Rudin, and Julie A Shah. 2014. The bayesian case model: A generative approach for case-based reasoning and prototype classification. *Advances in neural information processing systems*, 27.

Nathan Lambert, Thomas Krendl Gilbert, and Tom Zick. 2023. The history and risks of reinforcement learning and human feedback. *arXiv e-prints*, pages arXiv–2310.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Kellie Lu, Thomas Mesnard, Colton Bishop, Victor Carbune, and Abhinav Rastogi. 2023. Rlaif: Scaling reinforcement learning from human feedback with ai feedback. *arXiv preprint arXiv:2309.00267*.

Yuxi Li. 2017. Deep reinforcement learning: An overview. *arXiv preprint arXiv:1701.07274*.

Jinlu Liu, Liang Song, and Yongqiang Qin. 2020. Prototype rectification for few-shot learning. In *Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part I 16*, pages 741–756. Springer.

Yao Ming, Panpan Xu, Huamin Qu, and Liu Ren. 2019. Interpretable and steerable sequence learning via prototypes. In *Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pages 903–913.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, et al. 2021. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*.

OpenAI. 2024. Chatbot interaction for textual analysis and assistance. Conversational interactions with OpenAI's ChatGPT for generating text and providing language model assistance.

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.

Yingwei Pan, Ting Yao, Yehao Li, Yu Wang, Chong-Wah Ngo, and Tao Mei. 2019. Transferrable prototypical networks for unsupervised domain adaptation. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2239–2247.

Romain Paulus, Caiming Xiong, and Richard Socher. 2017. A deep reinforced model for abstractive summarization. *arXiv preprint arXiv:1705.04304*.

Shashi Kant Singh, Shubham Kumar, and Pawan Singh Mehra. 2023. Chat gpt & google bard ai: A review. In *2023 International Conference on IoT, Communication and Automation Technology (ICICAT)*, pages 1–6. IEEE.

Jake Snell, Kevin Swersky, and Richard Zemel. 2017. Prototypical networks for few-shot learning. *Advances in neural information processing systems*, 30.

Feifan Song, Bowen Yu, Minghao Li, Haiyang Yu, Fei Huang, Yongbin Li, and Houfeng Wang. 2023. Preference ranking optimization for human alignment. *arXiv preprint arXiv:2306.17492*.

Nisan Stiennon, Long Ouyang, Jeffrey Wu, Daniel Ziegler, Ryan Lowe, Chelsea Voss, Alec Radford, Dario Amodei, and Paul F Christiano. 2020. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33:3008–3021.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, et al. 2023. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*.

Ben Wang and Aran Komatsuzaki. 2021. GPT-J-6B: A 6 Billion Parameter Autoregressive Language Model. `https://github.com/kingoflolz/mesh-transformer-jax`.

Binghai Wang, Rui Zheng, Lu Chen, Yan Liu, Shihan Dou, Caishuang Huang, Wei Shen, Senjie Jin, Enyu Zhou, Chenyu Shi, et al. 2024. Secrets of rlhf in large language models part ii: Reward modeling. *arXiv preprint arXiv:2401.06080*.

Xiting Wang, Xinwei Gu, Jie Cao, Zihua Zhao, Yulan Yan, Bhuvan Middha, and Xing Xie. 2021. Reinforcing pretrained models for generating attractive text advertisements. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, KDD '21, page 3697–3707, New York, NY, USA. Association for Computing Machinery.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023. Rrhf: Rank responses to align language models with human feedback without tears. *arXiv preprint arXiv:2304.05302*.

Rui Zheng, Shihan Dou, Songyang Gao, Yuan Hua, Wei Shen, Binghai Wang, Yan Liu, Senjie Jin, Qin Liu, Yuhao Zhou, et al. 2023. Secrets of rlhf in large language models part i: Ppo. *arXiv preprint arXiv:2307.04964*.

Zhenxun Zhuang, Mingrui Liu, Ashok Cutkosky, and Francesco Orabona. 2022. Understanding adamw through proximal methods and scale-freeness. *arXiv preprint arXiv:2202.00089*.

Daniel M Ziegler, Nisan Stiennon, Jeffrey Wu, Tom B Brown, Alec Radford, Dario Amodei, Paul Christiano, and Geoffrey Irving. 2019. Fine-tuning language models from human preferences. *arXiv preprint arXiv:1909.08593*.