Prototypical Reward Network for Data-Efficient RLHF

Jinghan Zhang[1], **Xiting Wang**[2*], **Yiqiao Jin**[3], **Changyu Chen**[2], **Xinhao Zhang**[1], **Kunpeng Liu**[1*]
[1]Portland State University, [2]Renmin University of China, [3]Georgia Institute of Technology
{jinghanz,xinhaoz,kunpeng}@pdx.edu
{xitingwang,chen.changyu}@ruc.edu.cn
yjin328@gatech.edu

Abstract

The reward model for Reinforcement Learning from Human Feedback (RLHF) has proven effective in fine-tuning Large Language Models (LLMs). Notably, collecting human feedback for RLHF can be resource-intensive and lead to scalability issues for LLMs and complex tasks. Our proposed framework **Proto-RM** leverages prototypical networks to enhance reward models under limited human feedback. By enabling stable and reliable structural learning from fewer samples, Proto-RM significantly enhances LLMs' adaptability and accuracy in interpreting human preferences. Extensive experiments on various datasets demonstrate that Proto-RM significantly improves the performance of reward models and LLMs in human feedback tasks, achieving comparable and usually better results than traditional methods, while requiring significantly less data. in data-limited scenarios. This research offers a promising direction for enhancing the efficiency of reward models and optimizing the fine-tuning of language models under restricted feedback conditions.

1 Introduction

Reinforcement Learning from Human Feedback (RLHF) effectively integrates meticulous human judgment with the model's capacity for large-scale data processing (Cortes et al., 2015; Bai et al., 2022a; Stiennon et al., 2020), enhancing language models' adaptability to human communication styles and preferences (Yuan et al., 2023). By utilizing Reinforcement Learning (RL) over supervised fine-tuning, RLHF captures the complexity of human language, involving emotions, context, and subtle linguistic differences (Ouyang et al., 2022; Wang et al., 2023) thereby offering enhanced adaptability and flexibility in human interactions.

The success of RLHF hinges on the quality of the reward model (Wang et al., 2024; Lee et al., 2023; Bai et al., 2022b; Gilardi et al., 2023), which guides the RLHF learning process, ensures its accuracy and efficiency (Ouyang et al., 2022), while preventing deviations from desired outcomes (Paulus et al., 2017). A deficient reward model, however, may learn a complex yet inaccurate error surface, leading the model to favor high-scoring yet erroneous solutions (Chen et al., 2019, 2023; Li, 2017; Yang et al., 2024). This overfitting can result in responses that maximize rewards but diverge from the actual objectives and human preferences (Wang et al., 2021a). Notably, effective RLHF typically requires extensive data (Sun et al., 2023; Zhang et al., 2024a,b).

This Work. To address these challenges, we integrate prototypical networks–instance-based algorithms that learn representative prototypes for similar examples to facilitate tasks like classifications or regression (Snell et al., 2017)–with the reward model for RLHF. Prototypical networks are particularly suitable for few-shot learning as they efficiently extract key features from limited samples for decision-making (Liu et al., 2020). By optimizing the embedding process in the reward model using prototypical networks, we enable the reward model to learn stable and reliable data representation structures with limited sample sizes. This method is especially suitable in enhancing the model's learning and generalization from human feedback samples, given the limitations of sample quantity and the complexity of human preferences (Bai et al., 2022b).

To enhance the effectiveness of the reward model with limited human feedback data, we propose the **Proto**typical **R**eward **M**odel (**Proto-RM**) that decreases reliance on human feedback without compromising the performance of the reward model. The fundamental principle of the reward model is to assimilate human feedback to evaluate and steer the model outputs to meet human expectations and

[*]Corresponding Authors

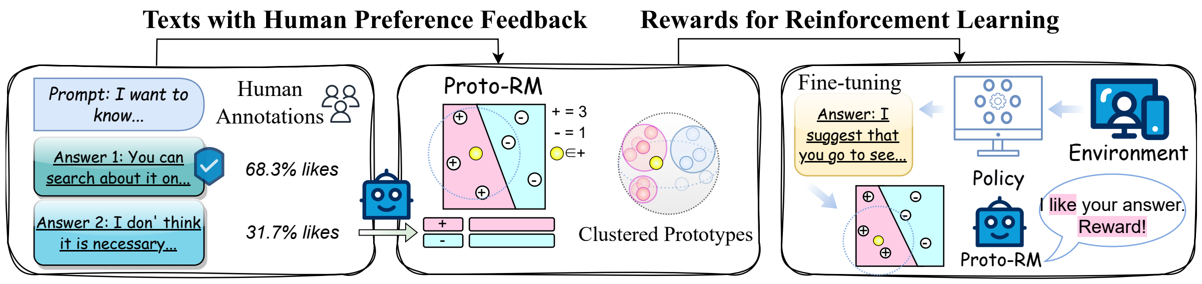
Texts with Human Preference Feedback
Rewards for Reinforcement Learning
Prompt: I want to know...
Human Annotations
Answer 1: You can search about it on...
68.3% likes
Answer 2: I don' think it is necessary...
31.7% likes
Proto-RM
+ = 3
- = 1
∈+
Clustered Prototypes
Fine-tuning
Answer: I suggest that you go to see...
Policy
Environment
I like your answer. Reward!
Proto-RM

Figure 1: Our proposed Proto-RM framework enhances reward model with prototypical networks. **Left**: Humans annotate pairwise RLHF data and select their preferred text. **Middle**: Proto-RM aggregates similar examples from embedding space into prototypes. **Right**: The enhanced reward model fine-tunes a pretrained LLM.

standards. Its crucial capability lies in effectively learning and extracting vital parameter information from limited human feedback, thus guiding the model's behavior. Our approach focus on learning a method that performs well in few-shot scenarios and is suitable for learning from human feedback samples while preserving the reward model's network structure and enhancing the parameterization capabilities.

Our method consists of three pivotal steps. First, in **Sample Encoding and Prototype Initialization**, we employ a reward model to encode samples. These encodings serve as the basis for initializing prototypes with a strategically selected subset of the encoded samples. Subsequently, we analyze the relationship between these initialized prototypes and the encodings of other samples. Second, in **Prototype Update and Addition**, we continuously refine the sample encodings based on their distances to the prototypes. Concurrently, we update the reward model's parameters by validating the predictions generated through the refined encodings. This iterative refinement ensures that prototypes accurately mirror the attributes of samples, enhancing the learning efficacy through human feedback samples. Finally, **Reward Model Fine-tuning** employs the refined prototypes and encodings to train the reward model. This training aims to accurately evaluate and guide the outputs of the language model, thereby improving the performance of LLMs during the fine-tuning stage.

Contributions. Our main contributions include:

- We propose Proto-RM, a novel prototypical-network-based method to improve the reward model. This structure facilitates training with fewer human feedback samples without compromising the learning ability of the reward model in scenarios with ample samples.
- We explore a prototypical learning method for human feedback samples, effectively managing human feedback that is difficult to quantify and varies in length.
- 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. Our experiments show that our method has clear advantages and achieves the effectiveness of training with extensive data, even when using limited samples.

2 Related Work

2.1 Reinforcement Learning from Human Feedback (RLHF)

Large Language Models (LLMs) such as GPT-4 (Achiam et al., 2023), Bard (Singh et al., 2023), and LLaMA-2 (Touvron et al., 2023) have demonstrated significant capabilities in understanding human languages and preferences (Zhao et al., 2024; Jin et al., 2024a,b). The efficacy of these models primarily hinges on Reinforcement Learning from Human Feedback (RLHF) (Christiano et al., 2017; Ziegler et al., 2019; Ouyang et al., 2022; Casper et al., 2023), which enables LLMs to iteratively refine their text generation capabilities, thereby producing outputs that accurately reflect human values and preferences (Song et al., 2023; Wang et al., 2021b). RLHF involves three interconnected stages: feedback collection, reward modeling, and policy optimization. Initially, human annotators assess model outputs and provide feedback. This feedback is then utilized in the reward modeling process, where supervised learning is used to train a reward model that replicates human assessments (Lambert et al., 2023; Dong et al., 2019). Finally, during policy optimization, the model is fine-tuned to produce outputs that receive positive evaluations from the reward model (Zheng et al., 2023). RLHF excels at identifying "good" behaviors than other reward specification or learning methods. However, it faces significant challenges due to the large volume of human feedback data required, which can be costly and resource-intensive (Beeching et al., 2023; Yao et al., 2023b).

2.2 Prototypical Networks

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). Researchers have extensively 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; Yao et al., 2023a). 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 objective is to train a reward model capable of training a policy that generates high-quality texts, as evaluated by humans, using a constrained set of human-annotated data.

Input. The input of the reward model is a dataset $\mathcal{D} = \{(x_i, y_i^+, z_i^+, y_i^-, z_i^-)\}_{i=1}^N$ with N examples, where each example consists of a common post $x_i \in \mathbf{X}$ and two corresponding summaries $y^+, y^- \in \mathbf{Y}$. These summaries are distinguished by human preferences: y^+ is the preferred (*chosen*) response with annotation $z_i^+ \in \mathbf{Z}$, and y^- is the less preferred (*rejected*) response with annotation $z_i^- \in \mathbf{Z}$, where $\mathbf{Z} = \{\text{chosen}, \text{rejected}\}$.

Output. The outputs consist of 2 components:

- Predictive scores $s_{(x_i, y_i^+)}$ and $s_{(x_i, y_i^-)}$ for each example (x_i, y_i^+, y_i^-), indicating the model's assessment of the relative quality of the summaries;

- The reward model $f_\phi : \mathbf{X} \times \mathbf{Y} \to \mathcal{E}$, where $\mathcal{E}$ is the embedding space. f_ϕ includes an embedding function e_ϕ and an aligned linear scoring process.

4 Methodology

The primary objective is to train a reward model that predicts which answer $\{y_i^+, y_i^-\}$ is better according to human judgment, given a prompt x_i.

4.1 Reward Model with Prototypical Network

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

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

Here, $\mathbf{e}$ is the representation of the input pair (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 $\mathbf{e}_{(x_i, y_i^*)}$, where $y^* \in \{y^+, y^-\}$, 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 $|\mathbf{P}|$ prototypes in class $\mathbf{p}^*$, where $|\mathbf{P}|$ is the total number of prototypes of class '$*$'. The formula for updating the embedding is expressed as:

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

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

4.2 Reward Model with Prototypical Network

Prototype Initialization. During the initialization phase, our goal is to properly initialize two

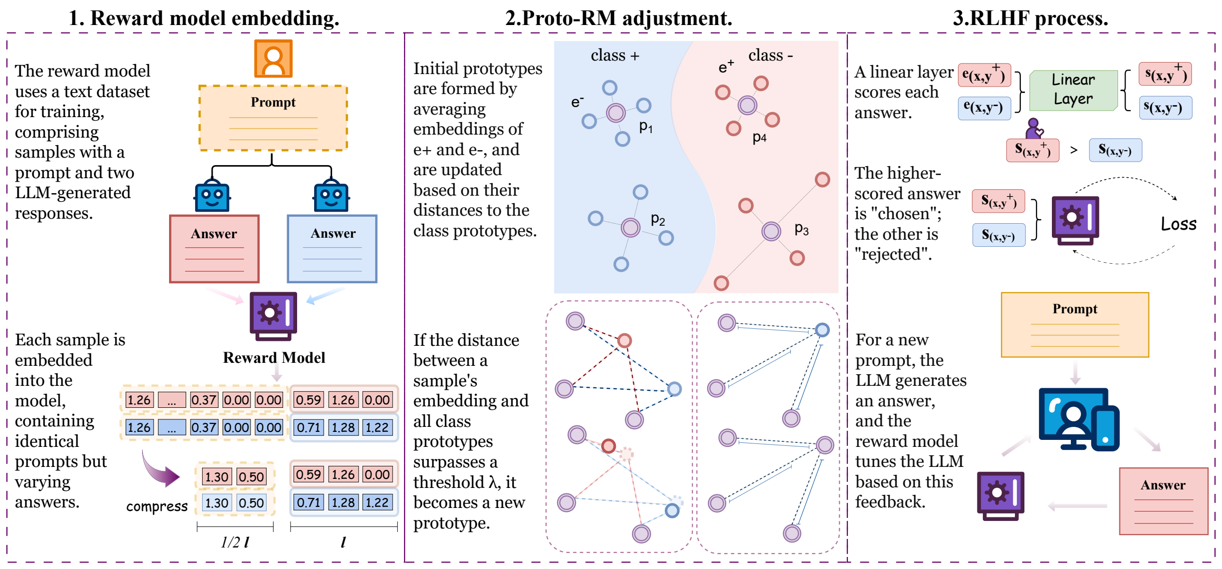
1. Reward model embedding.
2.Proto-RM adjustment.
3.RLHF process.
The reward model uses a text dataset for training, comprising samples with a prompt and two LLM-generated responses.
Prompt
Answer
Answer
Reward Model
Each sample is embedded into the model, containing identical prompts but varying answers.
1.26 ... 0.37 0.00 0.00 0.59 1.26 0.00
1.26 ... 0.37 0.00 0.00 0.71 1.28 1.22
compress
1.30 0.50 0.59 1.26 0.00
1.30 0.50 0.71 1.28 1.22
1/2 l
l
Initial prototypes are formed by averaging embeddings of e+ and e-, and are updated based on their distances to the class prototypes.
class +
class -
e-
e+
p1
p2
p3
p4
If the distance between a sample's embedding and all class prototypes surpasses a threshold λ, it becomes a new prototype.
A linear layer scores each answer.
e(x,y+)
e(x,y-)
Linear Layer
s(x,y+)
s(x,y-)
s(x,y+) > s(x,y-)
The higher-scored answer is "chosen"; the other is "rejected".
Loss
For a new prompt, the LLM generates an answer, and the reward model tunes the LLM based on this feedback.
Prompt
Answer

Figure 2: The framework consists of three components: 1) Reward model embedding, 2) Proto-RM 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.

classes of prototypes $\mathbf{p}_k \in \{\mathbf{p}^+, \mathbf{p}^-\}$. We randomly select n sample pairs and aggregate them according to their labels z_i. Specifically, we initialize different prototypes using the sample embeddings labeled "chosen" and "rejected". This strategy is employed to allow the model to better learn human preferences instead of mere differences in the content of the samples. We process the prompt and answer components of each text bar separately. For embeddings that initialize $\mathbf{p}_0$ to be the original prototypes , 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 consistently, reflecting a balanced representation of both prompt and answer components in the embedding space:

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

where $\text{align}(\cdot)$ denotes updating the embedding $\mathbf{e}_{x_i}$ to a new vector with the same maximum length as the longest embedding vector among all $\mathbf{e}_{x_i}$. Elements beyond the original length of $\mathbf{e}_{x_i}$ are padded with zeros. Similarly, we have:

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

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

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 using mean pooling as the aggregate function while keeping the parameters of the reward model ϕ frozen. During initialization, we disable gradient updates of the prototype vectors, ensuring that other model parameters do not affect the initialization process and thus guaranteeing its robustness.

Prototype Update. Our goal is to represent the samples effectively and comprehensively using prototypes. However, a fixed number of prototypes may not suffice for this purpose. Too few prototypes can lose important information, while too many can affect their representativeness and increase computational costs (Snell et al., 2017; Ming et al., 2019; Jin et al., 2023). Therefore, we consider employing Infinite Mixture Prototypes (IMP) (Allen et al., 2019) to automatically generate prototypes during training. This approach allows the model to increase the number of prototypes as needed, based on the distance relationship between the prototypes and the samples. The IMP technique is commonly used in classifying 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 text's abstract and multidimensional characteristics, 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 use the prototypical network to better assimilate new inputs. To enhance the representativeness and diversity of the prototypes, we 1) appropriately generate new prototypes and 2) continually update existing ones.

Generating New Prototypes. 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 (Wang and Blei, 2009). Our approach can balance between fitting simple data distributions with low capacity and complex distributions with high capacity.

We then compute the distance from each text bar in a text pair to every prototype $\mathbf{p}_k$ in their class, denoted as $d(\mathbf{e}_{(x_i,y_i^*)}, \mathbf{p}_k)$. Using the negative of these distances, we calculate the softmax to obtain the probability distribution of sample (x_i, y_i) belonging 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:

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

where ρ is the dropout ratio, K is the total number of prototypes, and $\lfloor \cdot \rfloor$ represents the floor function. Instead of random dropout, Proto-RM calculate the cosine similarity of prototypes within the same class and drop prototypes with the highest similarity. This approach ensures that the remaining prototypes are more diverse and thus more representative of the data. The new embedding $\mathbf{e}'_{(x_i,y_i^*)}$ is derived using the weighted average of $\mathbf{p}_k$ with respect to the probability distribution:

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

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. Inspired by Stiennon et al. (2020), we define the reward loss $\mathcal{L}_r$ as:

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

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 prototypes, we employ a hyperparameter τ to constrain their average in-between Euclidean distances. 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:

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

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

Algorithm 1 Reward Model with Prototypical Networks

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

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* (Pairwise) (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.

Implementation Settings. We use GPT-J (Wang and Komatsuzaki, 2021) as the pre-trained LLM. Our experiment applies 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. We use the AdamW optimizer (Zhuang et al., 2022) and search the best learning rate within the range of $[1e-6, 1e-5]$. Other hyperparameters are set to their default values as in Allen et al. (2019). We use the trlX framework (Havrilla et al., 2023) for model implementation. All experiments are conducted for a maximum of 5 epochs with early stopping on a server 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 ratios. 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, Proto-RM exhibits the most significant accuracy gain of 1.26%.

The line graphs Figure 3 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.

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

Table 2: Comparison of Proto-RM and Baseline across various datasets and sizes. Proto-RM consistently outperforms Baseline-RM in terms of accuracy.

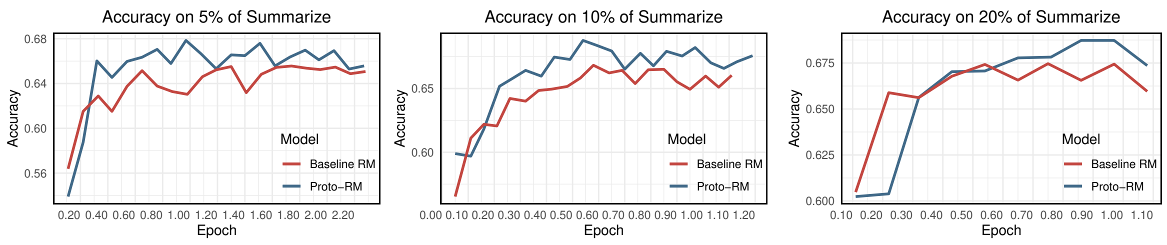

Figure 3: Comparison of reward models' accuracy on 5%, 10%, and 20% datasets.

	Toxicity		
RM	5%	10%	15%
Baseline	0.574	0.610	0.633
Proto-RM	0.588	0.625	0.654

Table 3: Toxicity levels for the Baseline and Proto-RM.

5.3 RLHF Performance

To ensure the consistency and integrity of the evaluation, we employ GPT-4 (OpenAI, 2024) to assess all outputs from GPT-J (6B) (Wang and Komatsuzaki, 2021) across four dimensions. This methodology aligns with various studies highlighting the capabilities of LLMs to produce high-quality text evaluation that align with or surpass human judgments (Gilardi et al., 2023; Alizadeh et al., 2023; Bai et al., 2022b). Our scoring criteria encompass factual accuracy, text relevance, information completeness, and clarity of expression, are uniformly applied. Each dimension receives a score up to 10, with increments of 0.5. The overall score is derived from the average of four metrics.

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 4 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. (Allen et al., 2019) 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 6 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 across epochs.

Effects of Dropout. As shown in Figure 5, 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.

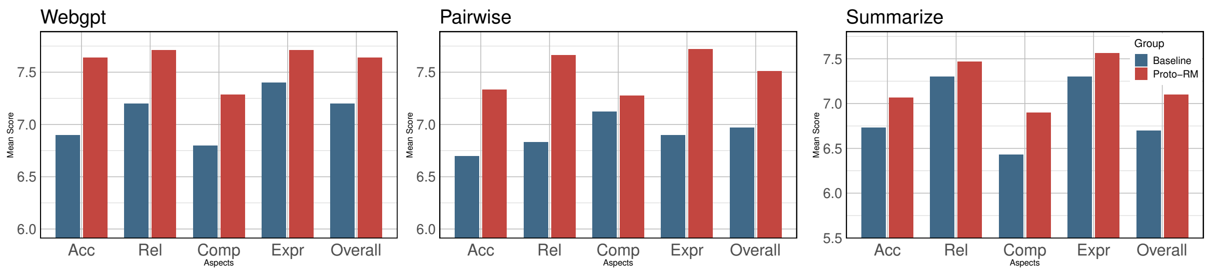

Figure 4: Performance of LLM with reward model fine-tuning.

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 4: Comparative responses from GPT-J models to a given prompt. The answer from the GPT-J with the Proto-RM is more coherent and relevant. GPT-J with Proto-RM catches all four necessary points in the text, and indicates a higher level of understanding and alignment with human preferences than the other models' responses.

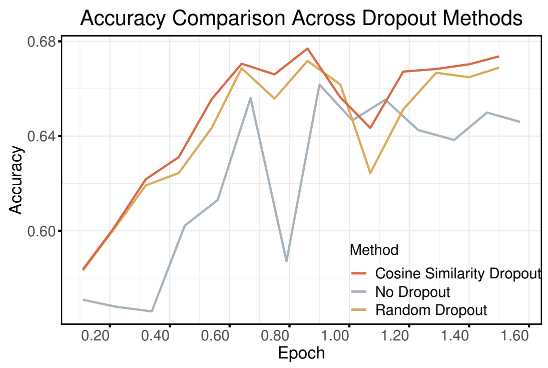

Figure 5: Impacts of Dropout. Models incorporating dropout exhibit higher accuracy, with Cosine Similarity Dropout performing slightly better than Random Dropout.

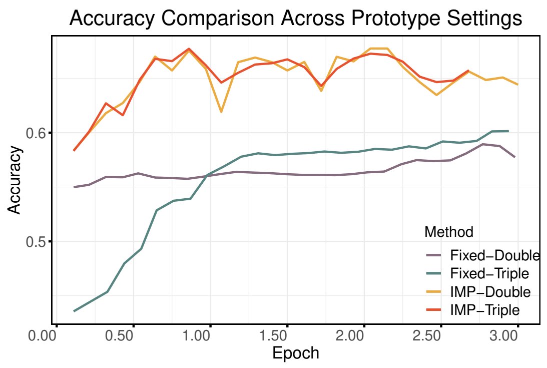

Figure 6: Impact of IMP. Models incorporating IMP demonstrate higher accuracy and more stable accuracy during training.

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.

Toxicity. We conduct experiments on different proportions of a toxicity dataset from (Bai et al., 2022a) over a single epoch. As shown in Table 3, our Proto-RM method outperforms the Baseline at random data proportions of 5%, 10%, and 20%.

6 Conclusion

Our research demonstrates the efficacy of prototypical networks in refining RLHF, 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. future works can explore the application of our method to more diverse and extensive datasets to further validate its effectiveness and adaptability.

Limitations There are a few limitations to the current framework. First, the Proto-RM performs best for open-domain pairwise human preference tasks that focus on alignment with human judgments. It remains unknown if Proto-RM yields substantial improvements in scenarios involving non=pairwise tasks or datasets that require less complex decision-making processes.

Second, we focus our study on the English-language human preference data. This source mainly represents the English-language NLP study. Our study does not include all the necessary research from other nature languages.

We examine how often NLP researchers cite older work by analyzing factors such as the mean age of citations. Our findings indicate a link between these factors and the frequency of citations. However, these links do not prove causation. More studies are needed to understand the reasons behind the citations of older papers.

Ethics Statement The Proto-RM framework can enhance the effectiveness and data efficiency of the RLHF process for LLM. However, as the framework uses pre-trained GPT-J as support and GPT-4 as an evaluator, it may inherit the ethical concerns associated with GPT-J and GPT-4, such as responding to harmful queries or exhibiting biased behaviors.

Acknowledgements This work was supported by Public Computing Cloud, Renmin University of China, Beijing Outstanding Young Scientist Program (BJJWZYJH012019100020098), Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China, and fund for building world-class universities (disciplines) of Renmin University of China. This work was partially done at Beijing Key Laboratory of Big Data Management and Analysis Methods, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing 100872, China, Public Policy and Decision-making Research Lab of RUC, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, MOE.

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*.

Changyu Chen, Xiting Wang, Yiqiao Jin, Victor Ye Dong, Li Dong, Jie Cao, Yi Liu, and Rui Yan. 2023. Semi-offline reinforcement learning for optimized text generation. In *ICML*.

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 *NeurIPS*.

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 *CIKM*, 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.

Yiqiao Jin, Mohit Chandra, Gaurav Verma, Yibo Hu, Munmun De Choudhury, and Srijan Kumar. 2024a. Better to ask in english: Cross-lingual evaluation of large language models for healthcare queries. In *The Web Conference*.

Yiqiao Jin, Minje Choi, Gaurav Verma, Jindong Wang, and Srijan Kumar. 2024b. Mm-soc: Benchmarking multimodal large language models in social media platforms. In *ACL*.

Yiqiao Jin, Xiting Wang, Yaru Hao, Yizhou Sun, and Xing Xie. 2023. Prototypical fine-tuning: Towards robust performance under varying data sizes. In *AAAI*.

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 *KDD*, 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 *CVPR*, 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.

Simeng Sun, Dhawal Gupta, and Mohit Iyyer. 2023. Exploring the impact of low-rank adaptation on the performance, efficiency, and regularization of rlhf. *arXiv preprint arXiv:2309.09055*.

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*.

Chong Wang and David Blei. 2009. Variational inference for the nested chinese restaurant process. *NeurIPS*, 22.

Xinlong Wang, Rufeng Zhang, Chunhua Shen, and Tao Kong. 2023. Densecl: A simple framework for self-supervised dense visual pre-training. *Visual Informatics*, 7(1):30–40.

Xiting Wang, Xinwei Gu, Jie Cao, Zihua Zhao, Yulan Yan, Bhuvan Middha, and Xing Xie. 2021a. Reinforcing pretrained models for generating attractive text advertisements. In *KDD*, KDD '21, page 3697–3707, New York, NY, USA. Association for Computing Machinery.

Xiting Wang, Xinwei Gu, Jie Cao, Zihua Zhao, Yulan Yan, Bhuvan Middha, and Xing Xie. 2021b. Reinforcing pretrained models for generating attractive text advertisements. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery & Data Mining*, pages 3697–3707.

Weikai Yang, Mengchen Liu, Zheng Wang, and Shixia Liu. 2024. Foundation models meet visualizations: Challenges and opportunities. *Computational Visual Media*, pages 1–26.

Jing Yao, Xiaoyuan Yi, Xiting Wang, Yifan Gong, and Xing Xie. 2023a. Value fulcra: Mapping large language models to the multidimensional spectrum of basic human values. *arXiv preprint arXiv:2311.10766*.

Jing Yao, Xiaoyuan Yi, Xiting Wang, Jindong Wang, and Xing Xie. 2023b. From instructions to intrinsic human values–a survey of alignment goals for big models. *arXiv preprint arXiv:2308.12014*.

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*.

Jinghan Zhang, Xiting Wang, Weijieying Ren, Lu Jiang, Dongjie Wang, and Kunpeng Liu. 2024a. Ratt: Athought structure for coherent and correct llmreasoning.

Xinhao Zhang, Zaitian Wang, Lu Jiang, Wanfu Gao, Pengfei Wang, and Kunpeng Liu. 2024b. Tfwt: Tabular feature weighting with transformer. *arXiv preprint arXiv:2405.08403*.

Qinlin Zhao, Jindong Wang, Yixuan Zhang, Yiqiao Jin, Kaijie Zhu, Hao Chen, and Xing Xie. 2024. Competeai: Understanding the competition behaviors in large language model-based agents. In *ICML*.

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*.

A Supplementary Statistics and Plots

A.1 Data Efficiency

In this section, we present a comparative analysis of our Proto-RM method against the Baseline results obtained from the standard RLHF on the full-size dataset. Notably, Proto-RM demonstrates equivalent effectiveness using only 20% of the data that the Baseline requires 100% to achieve. This significant reduction in data usage without compromising performance underlines the data efficiency of our method.

Here is a detailed table of the performance across various datasets:

Datasets	Baseline 100%	Proto-RM 20%
Webgpt	60.17%	60.56%
Pairwise	99.89%	99.84%
Summarize	68.88%	68.72%

Table 5: Performance comparison of Baseline RLHF and Proto-RM method.

For *Webgpt* dataset the Proto-RM method slightly improves the performance, with a performance increase from 60.17% to 60.56%. This improvement, while modest, indicates that our approach can enhance model effectiveness even with reduced data input.

For *Pairwise* dataset, the Baseline achieves a near-perfect score of 99.89%. The Proto-RM's performance closely follows at 99.84%, which is remarkable given that it only uses a fraction of the data.

For *Summarize* dataset, the performance of our Proto-RM method is comparably high at 68.72%, closely trailing the Baseline's 68.88%. Although there is a minor decrease, it falls within the margin of error and showcases that our approach can maintain high levels of effectiveness in summarizing tasks.

In all instances, the Proto-RM method validates our claim of data efficiency. Our method achieves competitive or even better results than the Baseline with significantly less data. This result shows that Proto-RM is suitable for improving data efficiency when the data cost is high or the data is limited.

A.2 Human Evaluations

To avoid any form of tuning specifically tailored to GPT-4's evaluation patterns, we present results in Table 7 and Table 8 evaluated by humans, which

Method	Time
Baseline	1:32:11
Proto-RM	1:58:08

Table 6: Training and testing time for the Baseline and Proto-RM methods.

may slightly vary from GPT-4's, but also validate the effectiveness of our method in aligning with human preferences.

A.3 Time Complexity

Here we analyze the time cost of Proto-RM. For all samples within each minibatch, the prototype updates introduce a time complexity of $O(n)$, where n is the number of samples. For each sample, Proto-RM needs to identify the closest k prototypes out of all m prototypes to compute the output of the prototype network. Although the IMP method gradually increases the number of prototypes, we have set a cap on the maximum number of prototypes that can be added. This cap is defined as α times the initial number of prototypes, where α is a fixed constant. Here, computing the output of the prototype network for each sample incurs a time complexity of $O(km)$. Thus, the introduction of prototype networks results in a time complexity of $O(nkm)$.

This analysis indicates that including prototypical networks does not significantly elevate the overall complexity. Additionally, we provide an analysis of the time taken for training and testing on a dataset of 8,552 samples to support our assertion that Proto-RM does not considerably increase computational costs in Table 6. We believe that the added complexity is manageable and justified by the performance improvements, especially considering the high data costs in such scenarios.

Metrics	Human 1	Human 2	Human 3	Human 4	Human 5	GPT-4
Accuracy	6.61	6.56	6.52	6.82	7.14	6.73
Relevance	7.17	7.33	6.98	7.15	6.90	7.3
Completeness	6.41	7.10	6.79	6.42	7.28	6.50
Expression	7.33	7.33	6.99	7.45	7.40	7.24
Overall	6.88	7.08	6.82	6.96	7.18	6.94

Table 7: Human Annotations on Summarize from Feedback for Baseline Model.

Metrics	Human 1	Human 2	Human 3	Human 4	Human 5	GPT-4
Accuracy	7.02	7.28	6.80	7.21	7.34	7.07
Relevance	7.11	7.35	7.35	7.41	7.02	7.40
Completeness	6.62	7.24	7.08	6.71	7.71	6.93
Expression	7.33	7.35	7.49	7.47	7.69	7.57
Overall	7.02	7.31	7.18	7.20	7.44	7.24

Table 8: Human Annotations on Summarize from Feedback for Proto-RM model.