Combined Preference-Based & Absolute Reward Signals for RLHF Fine-tuning

Master Thesis

To obtain the degree Master of Science (M.Sc.)
in the program Web & Data Science

Submitted by
AmeerAli Khan

First examinier:	Jun.-Prof. Dr. Dennis Riehle (Institut für Wirtschaftsinformatik)
Second examinier:	Anna Wolters, M.Sc. (Institut für Wirtschaftsinformatik)

Koblenz, in February 2024

Any sufficiently advanced technology is indistinguishable from magic.

Arthur C. Clarke

Contents

Figures

Tables

Abbreviations

A2C	Advantage Actor-Critic
ANN	Artificial Neural Network
BCE	Binary Cross Entropy
C.I	Confidence Interval
CLM	Causal Language Model
CoT	Chain-of-Thoughts
CRS	Combine Reward Signal
CT	Conversation Tree
DSRM	Design Science Research Methodology
GSM8K	Grade School Mathematics
IQR	Inter Quantile Range
KL	Kullback–Leibler
LLM	Large Language Model
LM	Language Model
LORA	Low-Rank Adaptation
LR	Learning Rate
LSTM	Long Short Term Memory
ML	Machine Learning
MLM	Masked Language Model
MSE	Mean Square Error
NLP	Natural Language Processing
OASST	Open Assistant
PEFT	Parameter Efficient Fine-tuning Technique
PMP	Preference Model Pre-training
PPO	Proximal Policy Optimization
QA	Question and Answering
RbRL	Rating based Reinforcement Learning
RELU	Rectified Linear Unit
RL	Reinforcement Learning
RLHF	Reinforcement Learning from Human Feedback
RNN	Recurrent neural Network
S.E	Standard Error
SFT	Supervised Fine-tuning
SOTA	State-of-the-Art
ULMFit	Universal Language Model Finetuning

Symbols

$\bowtie$	Natural Join
Π	Projection
σ	Selection

Zusammenfassung

Jüngste Studien haben gezeigt, dass die Feinabstimmung eines großen Sprachmodells mit Hilfe von Reinforcement Learning from Human Feedback die Fähigkeit des Modells verbessert, Benutzeranweisungen zu befolgen. Bei diesem Ansatz wird ein Belohnungsmodell eingesetzt, um das Sprachmodell so zu steuern, dass es Antworten erzeugt, die den menschlichen Präferenzen entsprechen. Traditionell werden diese Belohnungsmodelle mit Hilfe von Präferenz-Ranglisten trainiert, die sich ausschließlich auf relative Vergleiche konzentrieren und objektive Messungen der individuellen Antwortqualität nicht berücksichtigen. Der Einfluss von absolutem Feedback auf die Verbesserung der Textgenerierungsfähigkeit von Sprachmodellen ist in der aktuellen Literatur noch unerforscht. Diese Masterarbeit untersucht die Wirksamkeit der Einbeziehung von absoluten Belohnungssignalen in die Feinabstimmung des Reinforcement Learning from Human Feedback. Darüber hinaus wird der Kompromiss zwischen Präferenz und absoluter Belohnung untersucht, wenn diese miteinander kombiniert werden. Die Untersuchung zeigt, dass Modelle, die mit dem Modell der absoluten Belohnung trainiert wurden, besser abschneiden als solche, die mit dem Modell der Präferenzbelohnung trainiert wurden. Es wurde eine umgekehrte Korrelation zwischen dem Gewicht des Präferenz-Belohnungsmodells und der Leistung festgestellt, wenn die Belohnungswerte beider Modelle miteinander kombiniert werden. Das absolute Reward-Modell zeigt eine bessere Verallgemeinerbarkeit als das Preference-Reward-Modell, was seinen robusten Feedback-Mechanismus während der Feinabstimmung unterstreicht. Im Experiment erreichte das Modell, das mit dem absoluten und dem kombinierten Reward-Signal feinabgestimmt wurde, eine Gewinnrate von 66% bzw. 63% und übertraf damit die Modelle, die nur mit dem Präferenz-Feedback feinabgestimmt wurden.

Abstract

Recent studies have illustrated that fine-tuning large language model using Reinforcement Learning from Human Feedback enhances their capability to follow user instructions. This approach employ a reward model to guide language model toward generating responses that align with human preferences. Traditionally, these reward models are trained using preference rankings, focusing solely on relative comparisons and doesn't account for objective measure of individual response quality. The impact of absolute feedback in improving language model's text generation capability is unexplored area in current literature. This master thesis explores the efficacy of incorporating absolute reward signal into Reinforcement Learning from Human Feedback fine-tuning. Furthermore, it study the trade-off between preference and absolute reward when combined together. The investigation reveals that model trained with absolute reward model outperform those trained with preference reward model. It found inverse correlation between preference reward model weight and performance, when reward score of both models are combine together. The absolute reward model demonstrates better generalisability over the preference reward model, highlighting its robust feedback mechanism during fine-tuning. As per experiment, the model fine-tune with absolute and combined reward signal achieved win rate of 66% and 63% respectively, outperforming models fine-tuned solely with preference feedback.

1 Introduction

The Large Language Model (LLM) such as GPT3 (Brown et al. 2020) and LLAMA (Touvron et al. 2023b) are capable of State-of-the-Art (SOTA) performance on many Natural Language Processing (NLP) benchmarks and when prompted correctly they can perform a wide range of NLP tasks. However, these models tend to make up facts, occasionally fail to follow user instructions, and can produce impolite responses, making them less suitable for direct product application. Due to the training objective, the LLM models are difficult to control and bad at following human instructions. These models are designed to predict the next token in a sequence, which may not align to follow the user's instructions helpfully and safely (Ouyang et al. 2022).

To mitigate these challenges Ziegler et al. (2020) introduced an approach for fine-tuning LLM through Reinforcement Learning from Human Feedback (RLHF). The initial phase, SFT, guides LLM to follow instructions by learning from human-provided examples. This lays the groundwork for subsequent training steps. In a later step, the performance is further improved with Reinforcement Learning (RL) fine-tuning. This step incorporates human judgement into the reward signal to guide LLM toward generating not only accurate and polite but also genuinely helpful responses.

Typically, This RL fine-tuning process relies on a preference-based reward model that acts as a proxy for human judgments. Training data for this model are collected by presenting annotators with multiple responses for a single prompt and asking them to rank each response based on their informativeness and helpfulness. While the reward model trained using this method has succeeded in improving LLM to generate human-preferred responses, it inherently overlooks the individual quality of the response, potentially lacking objective feedback to the RL fine-tuning. Although the preference reward model provides a scalar reward score it is implicitly derived from the ranked data through a specific loss function. This reward model learns to provide scalar feedback by maximizing the likelihood that the reward score for a preferred response exceeds that of a less preferred one. The nuances of this technique and its implication for RLHF are further detailed in the 'Background' chapter.

In the field of RLHF, the existing methods for fine-tuning are centred on a preference reward model. The reliance on a preference reward model is evident in the seminal work of Christiano et al. (2023), who demonstrate the use of a reward model for complex RL environments. Following this, a body of research has focused on improving text generation through preference feedback provided by human annotators (Ziegler et al. 2020; Stiennon et al. 2022; Wu et al. 2023; Askell et al. 2021; Ouyang et al. 2022; Bai et al. 2022; Thoppilan et al. 2022). While this approach is widely used, White et al. (2023),

have highlighted that it lacks a global perspective on the quality of the individual sample. They argue that, since preference feedback is provided by pairwise comparison, it can only provide relative information and doesn't account for the absolute quality of each instance.

This methodology, although practical, fails to measure how good or bad the examples are. This mechanism only provides an option to choose the *best* among given and doesn't consider if all the given examples are bad. To address these limitations, White et al. (2023) introduce a novel Rating based Reinforcement Learning (RbRL) framework that approaches the reward learning tasks as a multi-class problem using a new multi-class cross-entropy loss function within a traditional RL setting. It asks human annotators to rate individual samples in the set $\{0, 1, \ldots, n-1\}$ indicating the absolute quality, where 0 is the lowest possible and $n-1$ is the highest possible rating.

Motivation: This evaluation by White et al. (2023) underscores a critical observation that has not been investigated, especially for complex tasks like text generation. The lack of empirical studies comparing the effectiveness of absolute reward signals alongside, or in combination with preference reward presents a significant gap. Motivated by the hypothesis that absolute reward may offer a direct measure of response quality. Which could enable a more precise alignment of LLM with human intention. Thus, this research aims to fill the gap by empirically investigating the impact of absolute reward signals on RLHF fine-tuning for text generation.

Our proposed work differs from the work of White et al. (2023) in the domain and methodology. We utilize the absolute feedback with continuous value and approach reward modelling as a regression problem instead of multi-class classification. Our reward modelling does not follow their framework and leverages a simple supervised learning algorithm. Our research evaluates the implication of using absolute reward signal on RLHF fine-tuning of LLM.

Research Objective: The purpose of this research is to study the impact of absolute reward on RLHF fine-tuning for text generation. It further investigates the impact of combining absolute reward alongside preference reward in RLHF finetuning. It examines the generalizability of both reward models and their capacity to provide consistent and robust reward signals during fine-tuning. It also does comparative analyses of the $rlhf$ model trained using these reward models. The objective is further detailed in the 'Methodologies' chapter.

This thesis contributes to the growing body of knowledge in RLHF fine-tuning by evaluating the impact of absolute reward signals and offering insights into the nuanced trade-offs between absolute and preference-based reward signals.

Constraints: The RLHF approach is resource-intensive and requires a vast amount of computation resources. Hence our research employs the Parameter Efficient Fine-tuning Technique (PEFT) as an alternative to full fine-tuning. This reduces the memory requirement and allows us to experiment on consumer-grade GPUs. Furthermore, due to time limitations, our research incorporates only a subset of the available data during the RL phase. Therefore results from our experiment should be interpreted within these constraints.

Thesis Structure: The master thesis is divided into seven chapters, starting with the Background, which lays the groundwork necessary to understand the proposed approach. It explains the foundation of the Machine Learning (ML) field and introduces the concepts of Artificial Neural Networks and RL. Then explain the theoretical details of the Language Model (LM) and Transformers (Vaswani et al. 2023) architecture used to train them. Finally, it provides a detailed understanding of the RLHF technique, a primary focus of this thesis.

The next chapter outlines the research methodology employed throughout the thesis, followed by its experiment detail, which delineates the setup used for experimentation. This setup chapter outlines the configuration and conditions under which the experiments were conducted. The Result and Discussion chapters present the outcomes of these experiments and evaluate whether the research objective has been achieved, offering insights and interpretation of the findings. Concluding the thesis, the final chapters summarize the research, acknowledging the limitations of the proposed approach. These sections also list down potential areas for future research, suggesting a direction for subsequent research to build upon our work.

2 Background

This chapter lays the groundwork essential to understanding the scope and depth of our research. It begins with the basics of ML and progressively builds to more complex topics like LM and diverse architectures used to train them. It further explains the fundamentals of RL and covers the more advanced topics in it such as the Advantage Actor critic method and Proximal Policy Optimization algorithm. Finally, it describes Reinforcement Learning from Human feedback (RLHF), a technique central to our research. It provides an in-depth discussion of the entire RLHF pipeline.

2.1 Fundamentals of Machine Learning

ML is a field of computer science which empowers machines to perform various tasks without being explicitly programmed. The field can be defined as the problem of improving an objective function through training experiences (Mahesh 2019). The objectives range from simple classification tasks such as classifying pictures of cats and dogs, to difficult tasks such as building an autonomous self-driving car. These tasks are learned by ML models, which act as parameterized functions with trainable parameters (Hunt et al. 2018). These parameters can be learned by optimization of an objective function when labels are available or by discovering the structure of the data when they are not. Labels are very important as they define tasks and determine the type of learning algorithm to be used for training the ML models. The algorithms are mainly categorized into supervised learning, unsupervised learning, self-supervised learning, and Reinforcement Learning (RL). Where RL is a type of learning where the model learns from trial and error. We will explore RL deeper in section 2.4, with its algorithms, applications, and its unique role in the broader field of ML.

Supervised Learning In, supervised learning, models are trained using labelled data. The trainable parameters are learned by mapping the input and output of the model from the given input-output pair (Mahesh 2019). In the context of supervised learning, the 'output' is called the 'target label' and the objective is to optimize the model to predict labels matching these target labels. The performance of the model is measured by the loss function which tells how far away the predicted labels are from the target label. The model is optimized by gradually minimizing this loss value.

Supervised learning tasks are mainly divided into two main types: classification and regression. If a task categorizes a given input into predefined classes, then it is known as a classification (Goodfellow et al. 2016). A commonly used classification algorithm is the Support Vector Machine. The task which predicts a continuous value based on the given

input is called the regression (Goodfellow et al. 2016). Linear Regression is a well-known algorithm for such tasks.

Unsupervised Learning Contrary to supervised learning, unsupervised learning doesn't have the advantage of learning from the target label. There is no incorrect answer because we don't have a target label to guide us (Mahesh 2019). The algorithm learns the structure of the input data and presents interesting insights into the data. The most popular use case of unsupervised learning is clustering. The clustering algorithm learns the structure of the data and organizes them into groups based on inter-cluster similarity and intra-cluster dissimilarity. K-means clustering is a widely employed algorithm for this purpose.

Self-Supervised Learning The self-supervised learning technique is a subset of unsupervised learning. It uses a unique approach to learning from unstructured data using supervised learning algorithms (Bergmann 2023). As discussed earlier, a supervised learning algorithm requires human-annotated data to learn from. However, this annotation is expensive to acquire. Hence, self-supervised learning techniques leverage the vast amount of unlabeled and unstructured data available from the internet and use the data itself to learn the representation from the unstructured dataset. The labels for unstructured data are created either by randomly masking a piece of information or by hiding the future sequence in a sequential dataset. the model learns to predict the missing information from a given context. This method is incredibly efficient because it doesn't rely on costly and labour-intensive labelled datasets (Bergmann 2023).

Model Training & Evaluation The training data is split into three parts, training set, validation set and optional but recommended test set. Typically, the training set consists of 80% of the data and the remaining data is divided into validation and test sets. The data is further divided into batches and then the algorithm is applied to each batch of the training set repeatedly (Goodfellow et al. 2016). After each iteration of training, the loss is calculated to optimize the weights of the model. Apart from optimizing the model parameter, we also need to tune the hyperparameter. The hyper-parameters are the configuration of the model and need to be tuned manually. An intuitive example of a hyper-parameter is the K-mean clustering algorithm where we manually decide the number of clusters the model should create. Once the model is trained, we can get predictions on the validation-set and test-set data. It is recommended that the test set is not involved in the training and hyper-parameter tuning process to avoid memorizing test data by the model (Goodfellow et al. 2016).

2.2 Artificial Neural Network

The Neural Network also known as Artificial Neural Network (ANN)s is a sub-field of ML. The architecture of ANN broadly follows the structure of the human brain and consists of thousands to millions of artificial neurons called nodes that are interconnected with each other (MIT 2017). Generally, the architecture is organized into layers of nodes taking input from the previous layer and forwarding their output to the next layer. Depending on the type of ANN the first layer either takes the text data in the form of embedding, image data in the form of RGB encoding, or simple numerical data. As illustrated in Figure 1 The output of each layer is passed to the subsequent layer as input and finally reaches the output layer to predict the values depending on tasks such as classification, regression, clustering etc.

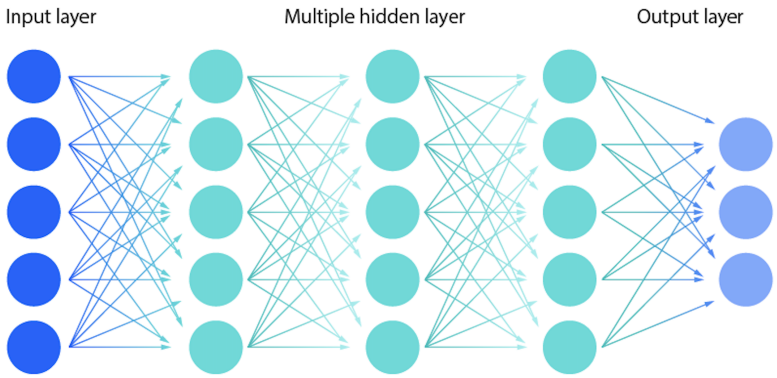


Figure 1 Simple architecture of Artificial Neural Network (IBM 2020)

Activation Function: Each node consists of linear operation followed by non-linearity. The linear operation is computed by taking a weighted average of inputs and then adding bias. The linear function is given by the equation: $\sum_i w_i x_i$ + bias (IBM 2020). Where x_i is each element of the input vector and w_i is its corresponding weightage.

The ANNs are set apart from other classical algorithms by their ability to learn from the non-linear data. This non-linearity is added by using a non-linear function. These functions are also called activation functions which transform the input from linear operation and decide if this input should be activated or not, meaning if a particular input is of importance or not. Two such popular activation functions are Rectified Linear Unit (RELU) and Softmax. RELU is a simple rectifier that outputs zero when values are below zero otherwise outputs back the original input (Goodfellow et al. 2016). It's a simple $\max(0, x)$ function and due to its simplicity, it's a widely used activation function in the hidden layer. On the contrary, the softmax activation is used in the output layer to normalize the predicted values. It converts raw scores (logits) into probabilities by taking the exponential

of each input value x_i and normalizing these values by dividing with the sum of all the exponentials. This activation is mainly used with classification tasks due to its nature of predicting the probability. The equation of softmax activation given by Goodfellow et al. (2016) is:

$$\text{softmax}(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}} \tag{2.1}$$

Loss Function: ANNs learn by iteratively optimizing the weights of the artificial neuron toward a specific goal. The objective of the ANN depends on the tasks and is generally defined using a loss function. The loss function calculates the difference between predicted values and target values. The loss function quantifies how far away the model is from the actual target value. The two most popular loss functions are Mean Square Error (MSE) and Binary Cross Entropy (BCE) used for regression and classification tasks respectively. The equation of both loss functions given by Goodfellow et al. (2016) are:

$$\text{MSE} = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2 \tag{2.2a}$$

$$\text{BCE} = -\frac{1}{n}\sum_{i=1}^{n}\left[y_i \log(\hat{y}_i) + (1 - y_i)\log(1 - \hat{y}_i)\right] \tag{2.2b}$$

MSE is a loss function measuring the average squared difference between the actual y_i and predicted values $\hat{y}_i$. It penalizes the large errors by squaring them. This leads the model to focus on minimizing those large discrepancies between predicted and actual outcomes. Whereas BCE quantifies the difference between the true labels and the predicted probabilities of the positive class. It penalizes incorrect predictions with a logarithmic function, making it highly effective for models outputting probabilities.

2.3 Language Model & Transformers

The LM is a statistical construct that determines the probability distribution over a sequence of words (Rosenfeld 2000). It is trained over the large corpus of text data available over the internet and books, allowing them to develop an understanding of the language structure, grammar, and semantics. Essentially, it gives the likelihood of a particular sequence of words occurring in a sentence. LM is an important tool in the field of NLP with various applications like speech recognition (Hannun et al. 2014), machine translation (Vaswani et al. 2023), text generation (Radford et al. 2019), text classification (Howard and Ruder 2018).

The statistical language model can be represented by the conditional probability of the word w_t, given all the preceding words in the sequence. However, when implemented for a large corpus it becomes computation-intensive and produces sparse probability distribution. Hence, in practice, the complexity of the problem is reduced by taking advantage of the fact that the probability of the word w_t is statistically more dependent on a closer word compared to all previous words (Bengio et al. 2000). This assumption underpins a simple but flawed implementation called the n-gram statistical language model and the most popular n-gram models are bi-gram and tri-gram.

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_t \mid w_{t-1}) \tag{2.3a}$$

$$P(w_1, w_2, \ldots, w_n) = \prod_{i=1}^{n} P(w_t \mid w_{t-2}, w_{t-1}) \tag{2.3b}$$

Figure 2 Equation for (a) bigram and (b) trigram statistical language model (Bengio et al. 2000)

The probability of a sequence of words $w_1, w_2, w_3, \ldots, w_n$ in the bigram model is calculated using the formula in Equation 2.3a. Where $P(w_t \mid w_{t-1})$ is the probability of the word w_t occurring immediately after the word w_{t-1}. Similarly, the trigram model extends this concept to two preceding words, as shown in Equation 2.3b. For both the bigram and trigram models, the conditional probability is estimated from a large corpus of text.

Word	**One-Hot Encoding**			
	London	**Delhi**	**Cat**	**Dog**
London	1	0	0	0
Delhi	0	1	0	0
Cat	0	0	1	0
Dog	0	0	0	1

Table 1 One-Hot encoded matrix for sample vocabularies

Another flaw of n-gram LMs is that it considers each word as a discrete independent entity, typically encoded as a one-hot vector, as illustrated in Table 1. While the numerical representation in the form of 0s and 1s is straightforward, it fails to capture the semantic relationship between words. For instance, in a given vocabulary, even though "London" and "Delhi" are both capital cities and "cat" and "dog" are animals, the one-hot encoding does not reflect the shared category. In response to this limitation, neural network LMs represent vocabulary in a continuous space where semantically similar words are posi-

tioned in proximity to each other (Arisoy et al. 2012). So "cat" would be closer to "dog" than to "London", reflecting an understanding of the semantics.

2.3.1 Neural Network Language Models

The neural network-based LM architecture proposed by Bengio et al. (2000) represents each word as a dense feature vector C, called word embedding. These embeddings are matrices of size $|V| \times m$, where $|V|$ is the vocabulary size and m is the embedding dimension, ranging from 30 to 300. Rather than relying on word frequency to calculate the conditional probability, this model estimates the probability via a neural network.

As illustrated in Figure 3, The model takes $t - n$ previous words as an input and predicts the word w_t's condition probability. It assigns each word $w_t - n$ an index of $w_t - n + 1$ and maps to its corresponding embedding vector. These embeddings are processed through the model's hidden layer and ultimately the output layer. Where the model estimates the conditional probability of w_t through an iterative process. Unlike statistical LM which stores a separate conditional probability of each set of context words, Neural network-based LM maintains a fixed set of word embedding used to estimate the conditional probability. These embeddings provide a richer meaning of words and their relation with the context words.

Since traditional neural network LM are not suitable for sequential processing, they had to be modified to use as a LM, limiting them to have a fixed set of context windows, meaning it only considers a limited number of previous words when predicting the next word (Goodfellow et al. 2016). To address this limitation, Recurrent neural Network (RNN) (Rumelhart et al. 1986) and Long Short Term Memory (LSTM) (Hochreiter and Schmidhuber 1997) were introduced to process sequential data such as time series, speech or text. This architecture allows storing and retrieving short-term and long-term information based on the context.

Theoretically, RNN (Rumelhart et al. 1986) can retrain information from all previous words, however, in practice, they often struggle due to issues like vanishing gradient, which makes the training process difficult. Hence, LSTM (Hochreiter and Schmidhuber 1997) were developed as an improvement of RNN, to process longer sequences without losing critical information. This architecture allows the model to control the flow of information by intelligently discarding past information and emphasizing critical information in current input.

2.3.2 Transformers

Despite their advantages both RNN and LSTM are computationally demanding. To address this, Vaswani et al. (2023) introduced an attention-based architecture called Transformer, which is less resource-intensive and outperforms RNN and LSTM in both performance and training speed. Transformers exclusively rely on the attention mechanism called self-attention. This mechanism allows for parallelization during training, which leads to faster training. Transformers have been break-through development of state-of-the-art sequence-to-sequence models, such as T5 (Raffel et al. 2023), which include both encoder and decoder, They have also enabled creation of encoder-only and decoder only model such BERT (Devlin et al. 2019), and GPT3 Brown et al. (2020) respectively. Showcasing usefulness across different applications.

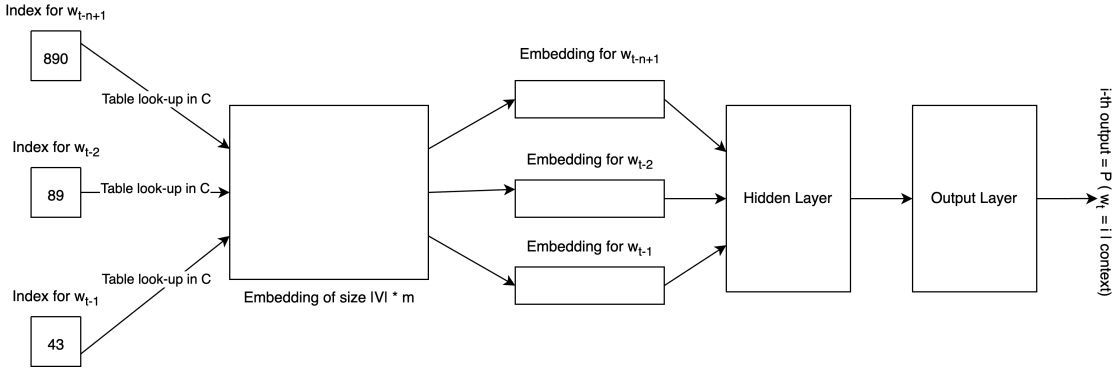


Figure 3 Architecture for neural language model (Bengio et al. 2000)

Encoder-Decorder Structure

Transformers utilize an encoder-decoder structure. Where the encoder encodes an input sequence into a vector representation of latent features. As illustrated in Figure 5, these representations are fed into the decoder to generate an output sequence. The decoder model operates auto-regressive, meaning at each step it takes all previously generated words as part of the input for generating the next word. For instance, in the translation of the sequence "Hi, how are you" to French, the encoder model will compress the sequence of words into fixed shape representations $z1, z2, z3$. Later, the decoder model uses this representation to generate the French translation one word at a time auto-regressively.

Vaswani et.al recommended stacking six identical layers for both encoder and decoder models. Each encoder layer comprises a multi-head attention layer and feed-forward network followed by layer norm. Whereas the decoder stack includes a masked multi-

head attention layer that prevents leaking of future information, followed by a standard multi-head attention layer and feed-forward network.

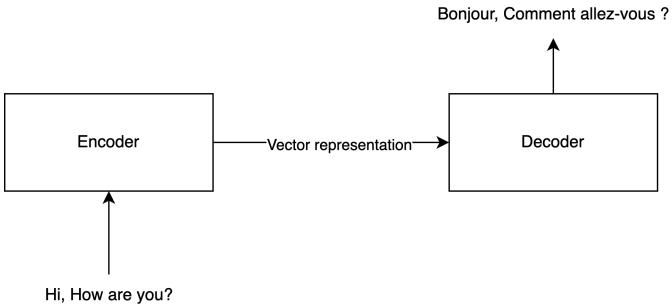


Figure 4 Basic encoder-decoder example

Language models that incorporate the Transformers architecture such as T5 (Raffel et al. 2023), BERT (Devlin et al. 2019), and GPT3 Brown et al. (2020) are known to stack a substantial number of layers leading them to have parameters ranging from million to hundreds of billion. The model with a considerably large number of parameters is known as LLM. This high magnitude of parameters in LLM allows them to have a deep understanding of language.

Positional Encoding

Sequential models like RNN and LSTM process inputs recurrently, one word at a time, allowing them to preserve the order of the sequence. However, Transformers feeds complete input simultaneously, Hence, a separate mechanism called "positional encoding" is introduced to preserve the order of the sequence. As proposed by Vaswani et al. (2023), the position encoding functions for sinusoidal encodings are defined as:

$$PE(pos, 2i) = \sin\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.4a}$$

$$PE(pos, 2i+1) = \cos\left(\frac{pos}{10000^{2i/d_{\text{model}}}}\right) \tag{2.4b}$$

They employed sinusoid functions of different frequencies for positional encoding, following a geometric progression from 2π to $1000 \cdot 2\pi$. In other words, these equations generate a unique value to identify the position of each word in a sentence. Where pos is the position of the word and i is the dimension. Since positional encoding is summed with word embedding, its dimension is the same as word embedding, d_{model}.

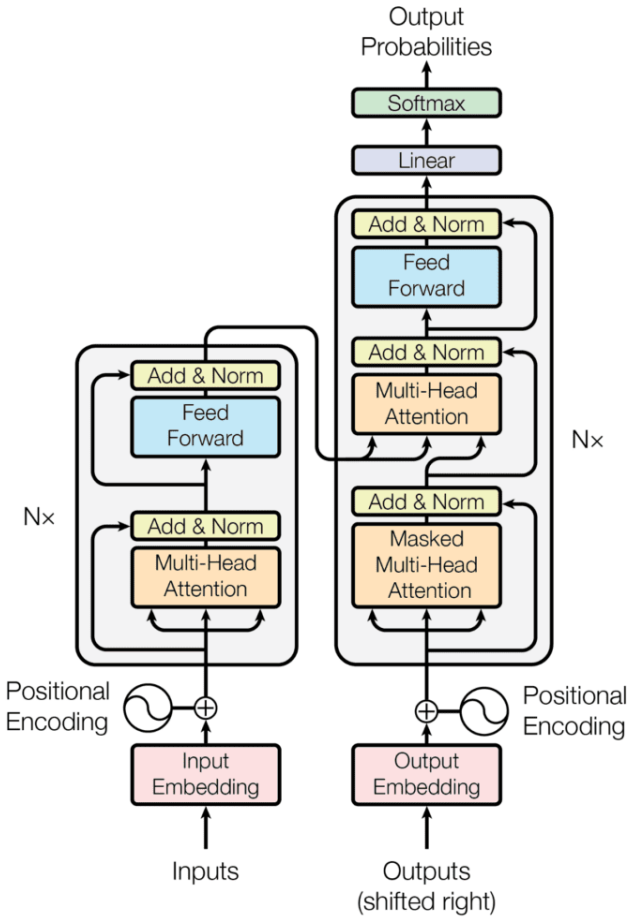


Figure 5 Architecture of Transformer based language model(Vaswani et al. 2023)

Attention Mechanism

As shown in Figure 4 encoder-decoder model compresses the source input into a fixed-sized vector representation, which can lead to potential information loss, especially for longer sentences and consequently deteriorates the performance. To address this problem Bahdanau et al. (2016) introduces the concept of 'attention' in encoder-decoder structure, as illustrated in Figure 6. Instead of compressing complete source input into one single vector representation, the model maintains a vector representation for each word in the sequence, allowing the model to account for individual words' importance. When the decoder generates a word, it 'searches' for the most relevant information with the magnitude of importance for each word in the source sentence indicated by the attention layer.

The shared attention between the encoder and decoder is termed cross-attention, while, self-attention refers to the mechanism used to understand the context within a single sentence. Self-attention in transformers allows each word in the sequence to account for its

relationship with every other word, thus allowing the model to contextually understand each word within the entire sentence (Vaswani et al. 2023).

Transformers calculate attention weights by using 'Scaled Dot-Product Attention'. As illustrated in Equation 2.5 and Figure 8 (Vaswani et al. 2023), Where Q, K and V is a query, key and value vector of dimension d_k. The dot-product of the query and key vector is taken before scaling it by d_k and applying the SoftMax function to obtain the weights for value. To enhance performance, the model computes multiple sets of these attentions known as multi-head attention and then concatenates them together.

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.5}$$

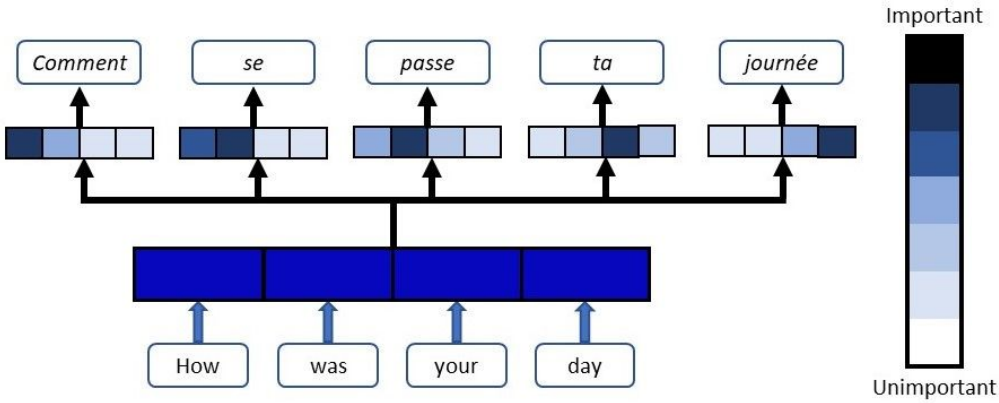


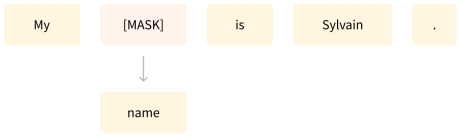
Figure 6 Example of attention mechanism (Loye 2019)

LM Pretraining & Fine-tuning

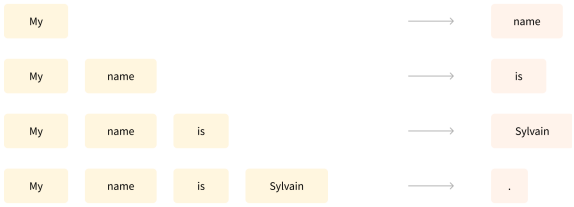
Pretraining is a foundation step in the development of machine learning models especially within the domain of NLP. While earlier NLP approaches utilized pre-trained word embedding layers, such as those proposed by Mikolov et al. (2013), they randomly initialized the parameter for remaining layers and trained them from scratch on the target tasks. Howard and Ruder (2018) advanced the field with the concept of Universal Language Model Finetuning (ULMFit), which significantly improves the effectiveness of transfer learning in NLP by training the model initially on large corpus and then fine-tuning on specialized tasks. Pretraining follows the principle of self-supervised learning, where several pretraining tasks are designed that enable the model to learn from a large corpus of general-domain text before fine-tuning specific tasks.

A notable pre-training technique for encoder-only models was introduced by Devlin et al. (2019), showcasing pre-training tasks like Masked Language Model (MLM). This tech-

nique employs a bidirectional model to predict words, that have been intentionally masked in the input sequence, by relying only on the context provided by its neighbouring words. However, because MLM allows the model to 'see' future words during pretraining it is not suitable for training traditional LM which requires sequential processing of the input. For that, Causal Language Model (CLM) is used, which processes input sequentially without 'peeking' into the future and trains the model to predict the next word given the preceding context. Both CLM and MLM are depicted in the Figure 7.



(a) Mask Language Model



(b) Causal Language Model

Figure 7 Pretraining tasks for language model (HuggingFace 2021)

Fine-tuning, an essential phase after pretraining, is a supervised learning technique to transfer the knowledge of a pre-trained model to solve a specific task. The model is initialized using the parameter weights of the pre-trained model and then trained toward a new objective using a labelled dataset. Such specific tasks, known as downstream tasks, enclose a broad spectrum of problems, including text classification, summarisation, generation, and question and answer.

2.4 Reinforcement Learning

RL is subfield of ML that focus on sequential decision-making tasks. In the RL ecosystem, the model is often referred to as an agent. These agents learn through trial and error by interacting with an environment, performing actions, and receiving corresponding rewards. Contrary to supervised learning, where the model relies on input-output pairs within a static dataset, RL, agents iteratively update their behaviour based on the reward they receive from the environment (Kaelbling et al. 1996). The agent receives positive reinforcement from the environment for desired actions and negative reinforcement for undesired ones.

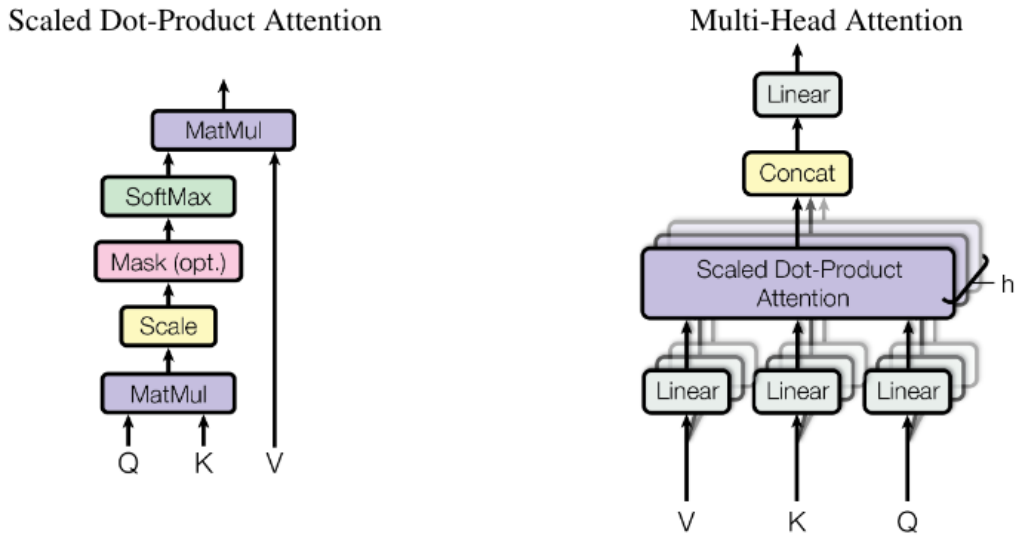


Figure 8 (left) Scaled Dot-Product Attention. (right) Multi-Head Attention (Vaswani et al. 2023)

As Goodfellow et al. (2016) explained, in RL, an agent interacts with the environment over a period. At each time step t, the agent observes the state s_t from the state space S and takes an action a_t from the action space A. The state s_t of the environment contains all information necessary for the agent to act. Subsequently, the environment provides a reward r_t. Considering the example in Figure 9, the agent is an autonomous player, and the game is its environment. The agent observes each frame of the game and takes an action. Upon taking the action at time step t, it receives a reward r_t and observes the new state $s_t + 1$ of the environment. The agent follows the policy $\pi(a_t \mid s_t)$, which maps its state-action pairs. Guiding the agent on the action to take given its current state. The goal is to learn the optimal policy π^* that maximizes the cumulative reward or expected return when followed.

2.4.1 Policy Function & Value Function

In RL, an agent can discover the optimal policy π^* either by directly learning the policy or by estimating the value function. If the policy is deterministic, then the agent learns to take specific action for each state. In contrast, if the policy is stochastic then the agent outputs a probability distribution over action for the given state. Figure 10a illustrates an example of a directly learned policy, where for each state the agent has a pre-determined action that leads to achieving maximum expected return. This type of function is known as the Policy Function (Goodfellow et al. 2016).

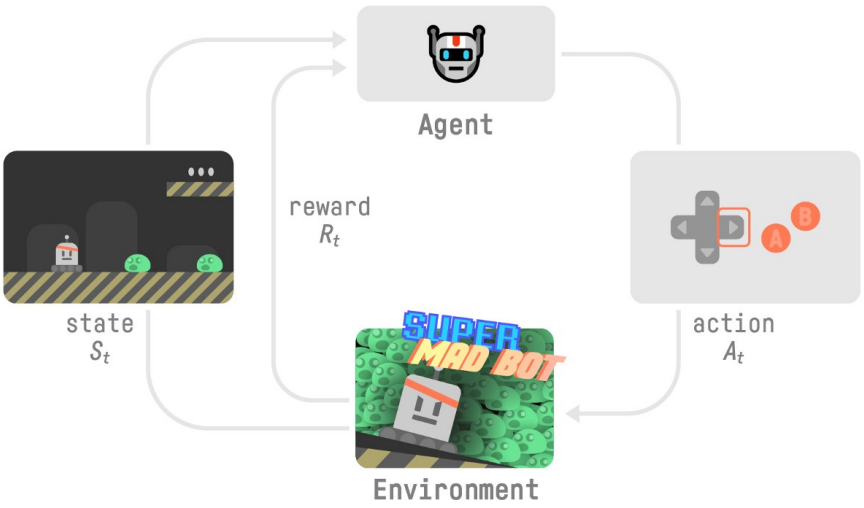


Figure 9 Example of a reinforcement learning environment (Simonini and Sanseviero 2023)

The Value Function offers an alternate strategy to the policy function. It estimates the 'worth' or 'value' of the given state. The value of the state is the expected return the agent gets when it starts in that state and follows the policy π afterwards, as illustrated in Figure 10b. In this case, the policy involves taking the action which leads to the state with maximum value (Li 2018). There are two types of value functions:

- **State Value Function** $V(s_t)$: It estimates the expected return of starting at state s_t and following the policy π afterwards.
- **Action Value Function** $Q(s_t, a_t)$: It estimates the expected return of starting at state s_t, taking an action a_t and then following a policy π afterwards.

Typically, methods that focus only on policy functions are called actor-only methods, while methods that rely only on value functions are known as critic-only methods (Goodfellow et al. 2016).

2.4.2 Policy Gradient

Policy gradient algorithms, as highlighted by Schulman et al. (2017), are considered critical algorithms within the category of the policy-based method. Diverging from value-based methods that focus on estimating value function, policy gradients directly optimize the policy function. The policy is often parameterized by neural networks, which are best at approximating complex, non-linear functions. The goal of the policy gradient method is to find parameters θ that maximize the expected return of the trajectory. This is achieved

by optimizing the objective function $J(\theta)$, as shown in Equation 2.6 (Simonini and Sanseviero 2023). The objective function $J(\theta)$ is defined as the expectation under the policy π_θ of the product of the probability of a trajectory τ given θ, and return $R(\tau)$ of that trajectory, summed over all possible trajectory τ.

(a) Policy Function **(b)** Value Function

Figure 10 An example of the agent following policy and value function (Simonini and Sanseviero 2023)

$$J(\theta) = \mathbb{E}_{\pi_\theta}\left[P(\tau;\theta)R(\tau)\right] \tag{2.6}$$

Optimization of $J(\theta)$ is achieved using a gradient ascent algorithm. The idea here is to move the values of θ in the direction that gives maximum expected return. This optimization process involves calculating the gradient of the objective function, however, calculating the true gradient of the $J(\theta)$ would require considering the probabilities of all possible trajectories τ, which is computationally expensive (Simonini and Sanseviero 2023). Therefore gradient of the objective function is estimated over a limited number of sampled trajectories. The policy gradient theorem, as described in Equation 2.7 (Simonini and Sanseviero 2023), provides an approach to estimate the gradient. It is defined as the expected value over a distribution of trajectories, of the product of the gradient of the logarithm of the policy π_θ at a given state-action pair (s_t, a_t) and the return $R(\tau)$.

$$\nabla J(\theta) = \mathbb{E}_{\pi_\theta}\left[P(\tau;\theta)R(\tau)\right] \tag{2.7}$$

The policy gradient algorithm iteratively updates the policy π_θ. It samples a limited number of trajectories using the current policy π_θ, calculates the gradient of the objective function based on these sampled trajectories and then updates the policy using gradient ascent, as indicated in Equation 2.8 (Simonini and Sanseviero 2023). The scoring function $R(\tau)$ increases the probability of taking the action a_t in the given state s_t if the return is high otherwise decreases the probability.

$$\theta = \theta + \alpha \hat{g} \quad (3) \tag{2.8}$$

2.4.3 Advantage Actor Critic

The Advantage Actor-Critic (A2C) approach refines the policy gradient methodology by introducing a mechanism to reduce variance in the estimation of $\nabla J(\theta)$. As shown in Equation 2.7 (Simonini and Sanseviero 2023), to compute this gradient, it is necessary to calculate $R(\tau)$ of the trajectory τ. However, due stochastic nature of the environment and policy, trajectories from the same starting state can end differently, leading to varying returns, resulting in high variance (Simonini and Sanseviero 2023). To reduce the variance large number of trajectories are needed, which reduces the sample efficiency of the method. This variability creates problems and leads to unstable learning

$$A(s_t, a_t) = Q(s_t, a_t) - V(s_t) \quad (4) \tag{2.9}$$

To address this challenge, A2C utilize the actor-critic method, where the agent's actions are directed by an *Actor*, which operates on a policy function, while the quality of these actions is assessed by a *Critic*, using a value-based method (Simonini and Sanseviero 2023). A2C introduced an advantage function as a key component of the critic. As shown in Equation 2.9 it measures the relative advantage of taking an action a_t in the s_t against the average value of that state. By focusing on the advantage, the A2C method seeks to normalize the return and thus, reduce the variance associated with them. This normalization process allows the policy updates to be more stable and can improve learning efficiency. The updated equation for the policy gradient, incorporating the advantage function is shown in Equation 2.10 (Schulman et al. 2017)

$$\nabla_\theta J(\theta) = \tilde{g} = \mathbb{E}_t \left[\nabla_\theta \log \pi_\theta(a_t \mid s_t) A(s_t, a_t) \right] \quad (5) \tag{2.10}$$

In practice, this means that A2C algorithm increases the probability of selecting the actions that have a higher than average advantage, and conversely, decreases the probability of selecting the actions with a lower than average advantage (Simonini and Sanseviero 2023). With this method, the A2C update the policy more efficiently.

2.4.4 Proximal Policy Optimization

Traditional policy gradient methods, while effective in learning policies, often suffer from high variance and instability, especially in complex environments. Although incorporating an actor-critic framework with the policy gradients method reduces the variance and makes the training process more robust, challenges persist. However, these methods could still encounter issues with large policy updates, deviating too much from optimal policy, particularly in complex environments (Simonini and Sanseviero 2023). Proximal Policy Optimization (PPO), developed by Schulman et al. (2017) addresses these issues with a data-efficient and robust approach. PPO introduced a novel approach to constrain the policy update through a new objective function called the Clipped surrogate objective function, which limits the policy update within a small range by clipping the ratio of the new policy probability to the old policy probability.

Clipped Surrogate Objective

The Clipped Surrogate Objective function is illustrated in Equation 2.11(Schulman et al. 2017), where ϵ is a tunable hyperparameter. The objective function's first component is similar to Equation 2.10, comprising the product or ratio and *advantage*. The ratio is the likelihood of taking an action a_t in state s_t under the current policy, normalized by its probability under the previous policy. The second component is the constrained version of the first term, where the ratio is clipped with the predetermined range. This clipping mechanism reduces the magnitude of policy updates. It ensures that the new policy is not too different from the old one. It promotes small incremental changes by taking the minimum value between the clip and the unclipped objective. This allows controlled policy updates and stable learning through the training phase.

$$L^{\text{CLIP}}(\theta) = \hat{\mathbb{E}}_t \left[\min \left(r_t(\theta)\hat{A}_t, \text{clip}\left(r_t(\theta), 1-\epsilon, 1+\epsilon\right)\hat{A}_t \right) \right] \tag{2.11}$$

2.5 Reinforcement Learning from Human Feedback

Pretrained LLMs have shown impressive capabilities in the field of NLP to perform a range of tasks. However, since these models are trained to predict the next word of the input sequence of a document sourced from the internet, resulting to often express undesirable behaviours such as fabricating facts, generating offensive responses, or simply failing to follow users' instructions effectively and safely (Ouyang et al. 2022). Thus, indicating that the training objective of LLMs are misaligned. To mitigate these problems Ouyang et al. (2022) and Askell et al. (2021) utilized a fine-tuning process known

as RLHF, to align general purpose LLMs with user intention more closely. This training method is designed to help the model assist users in solving tasks without misleading them by making up facts or causing psychological or social harm.

Ziegler et al. (2020) were among the first to employ RLHF in NLP tasks. They pioneered a novel approach to fine-tuning LLMs. Previously, algorithmically defined reward functions like BLEU (Papineni et al. 2002) or ROGUE (Lin 2004) were used for RL fine-tuning. Ziegler et al. (2020), however, trained a reward model (also known as preference model) using binary feedback to fine-tune LLM to generate text in positive and descriptive style and to produce human-preferred summaries of long text. Later, Stiennon et al. (2022) confirmed that optimising directly for the reward model yields better summarization results compared to optimizing the ROGUE score.

The reward signal is a critical aspect of RL, it's straightforward to design reward functions for tasks like chess or maze navigation. Where positive feedback is given when the agent wins the game or reaches the goal, and negative feedback when it loses the game or is stuck in the maze. However, for tasks like social media moderation or creative storytelling, the reward function is difficult to design due to subjective interpretation. Whereas it is feasible to provide the agent with human feedback whether the task outcome is desirable or not. In other words, aligning the outcome of the model with human values by providing human feedback.

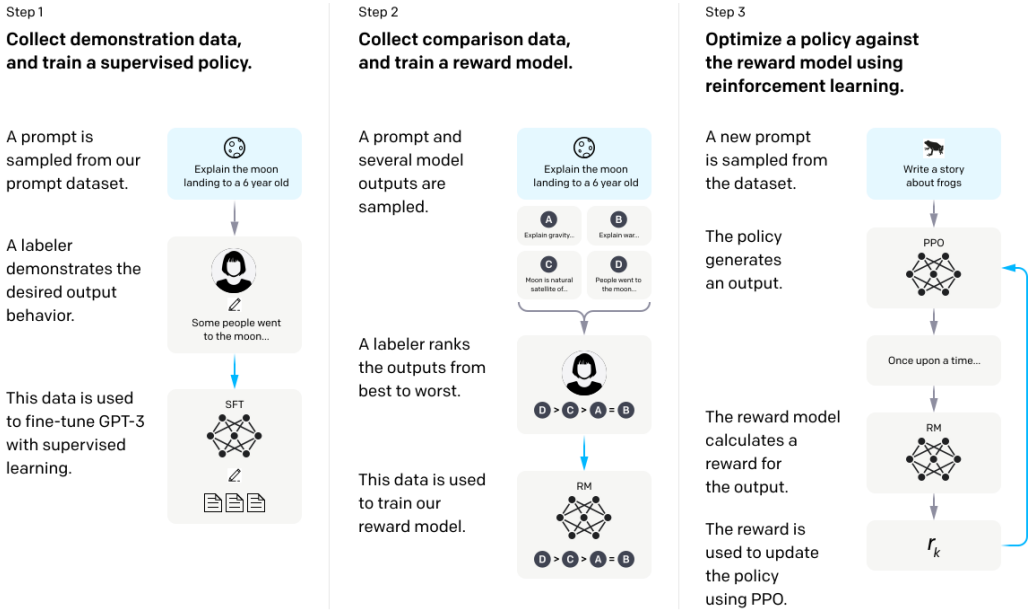


Figure 11 Three steps of RLHF pipeline(Ouyang et al. 2022)

Human evaluation done on the model outputs shows that the 1.3 billion parameter model is preferred over the 175 billion parameter model despite having a hundred times fewer

parameters, due to its ability to align more closely with human values (Ouyang et al. 2022). The RLHF fine-tuning technique has proven to reduce the toxicity and hallucination in models' response and facilitate the alignment of LLM with human values (Askell et al. 2021; Bai et al. 2022; Ouyang et al. 2022). The steps involved in RLHF training pipeline is illustrated by Figure 11.

2.5.1 Data Collection & Initial Policy

For RLHF training it's important to demonstrate desired behaviour to the model. This demonstration serves as a starting point for our model. A pre-trained LLM is chosen, and a list of prompts is compiled to collect human demonstration. Trained labellers were asked to provide to demonstrate the 'preferred' response using which the supervised fine-tuning process is performed. This fine-tuning process is called imitation learning, where the model learns to mimic high-quality responses (Ouyang et al. 2022). The Supervised Fine-tuned (SFT) model serves as the basis for subsequent training stages.

2.5.2 Reward Model Training

Performing supervised fine-tuning with demonstration alone can result in a model that does not adequately capture the broader intentions of humans, both explicit and implicit. Therefore, RL fine-tuning is done with human feedback as a nuanced reward signal. However, direct application of such feedback is resource-intensive, requiring substantial human involvement. Especially for a system which requires hundreds or thousands of hours of experience. To practically fine-tune the LLM using human feedback, it's crucial to significantly reduce the volume of feedback needed (Christiano et al. 2023). Hence, to manage the resource-intensive nature of direct feedback, there is a need for a reward model which acts as a proxy for human feedback. This model assesses generated text and numerically indicates human preference with a scalar reward.

To create the training data for reward modelling, responses are generated using the SFT model for the sampled prompt from the predefined dataset. Either K responses for each prompt are generated using a single SFT model or multiple models are used to generate responses on the same prompt. Later, trained human labellers are asked to rank these prompt-responses pairs in descending order as per their preference.

Finally, a reward model is trained using preference-ranked data, the model takes prompt and its responses as input and generates a scalar reward as output to accurately reflect the human preference. An optional step is to initialize the reward model with SFT or train it from scratch. The intuition behind using the same SFT model is that the reward

model should have a similar capacity of understanding the text to the model which is generating them (HuggFace 2023). In this case, the final embedding layer of the SFT model is replaced with a linear layer to output a scalar reward score.

$$\text{loss}(\theta) = -\frac{1}{\binom{K}{2}}\mathbb{E}_{(x,y_w,y_l)\sim\mathcal{D}}\left[\log\left(\sigma\left(r_\theta(x,y_w) - r_\theta(x,y_l)\right)\right)\right] \tag{2.12}$$

The reward model is trained using an objective to optimise preference prediction. The loss function of this objective is formulated by Equation 2.12. For K responses this produces $\binom{K}{2}$ comparison for each prompt shown to the labeller. Training is conducted with each batch containing all possible $\binom{K}{2}$ combinations from each prompt, and loss is average over these comparisons. In these pairs, y_w and y_l are respective preferred and rejected (or less preferred) responses for the prompt x, whereas $r_\theta(x, y)$ is a scalar output of reward model with prompt x and response y as input (Ouyang et al. 2022). The objective of the model is to provide scalar scores such that the value for the preferred response is higher than that of a non-preferred response.

Preference Model Pretraining

The reward model which aligns well with human preference can act as a human proxy in RLHF fine-tuning process to provide feedback. However, to train these reward models expensive high-quality human interaction is needed. Thus, to improve the sample efficiency of the reward model Askell et al. (2021) explored Preference Model Pre-training (PMP), which follows the sequential training procedure as:

Language Model Pre-training → Preference Model Pre-training → Preference Model Finetuning

The reward model is initialized with the pre-trained model or an optional SFT model and employs large-scale public datasets like Stack Exchange and Reddit, repurposing user interaction for training. The dataset consists of user interaction on various topics among multiple users in the form of questions and answers. Answers with the highest user ratings are considered the most preferred, similarly, other rank responses are considered based on the received votes. This training process has been proven to improve sample efficiency and often improves the performance of reward models when later fine-tuned with high-quality human feedback.

2.5.3 Reinforcement Learning Fine-tuning

RL fine-tuning is a crucial step in the RLHF pipeline. Here, models trained in earlier steps work together to guide the generated response toward human preference. HuggFace (2023) explains how the LLM fine-tuning task is formulated as an RL problem, with the language model as a policy that takes in a prompt and outputs a sequence of text. The policy's action space comprises all the tokens corresponding to the vocabulary of the language model and the observation space encompasses the distribution of all possible input token sequences, where the dimension of the observation space is as same as the dimension of vocabulary multiplied by the number of input tokens. Finally, the reward function combines a scalar score from the reward model and a constraint on policy shift. An SFT model copy is used as an initial policy for finetuning. This policy continuously updates iteratively, so it is referred to as 'active policy', while another copy is stored as a base policy to calculate the policy shift (HuggFace 2023).

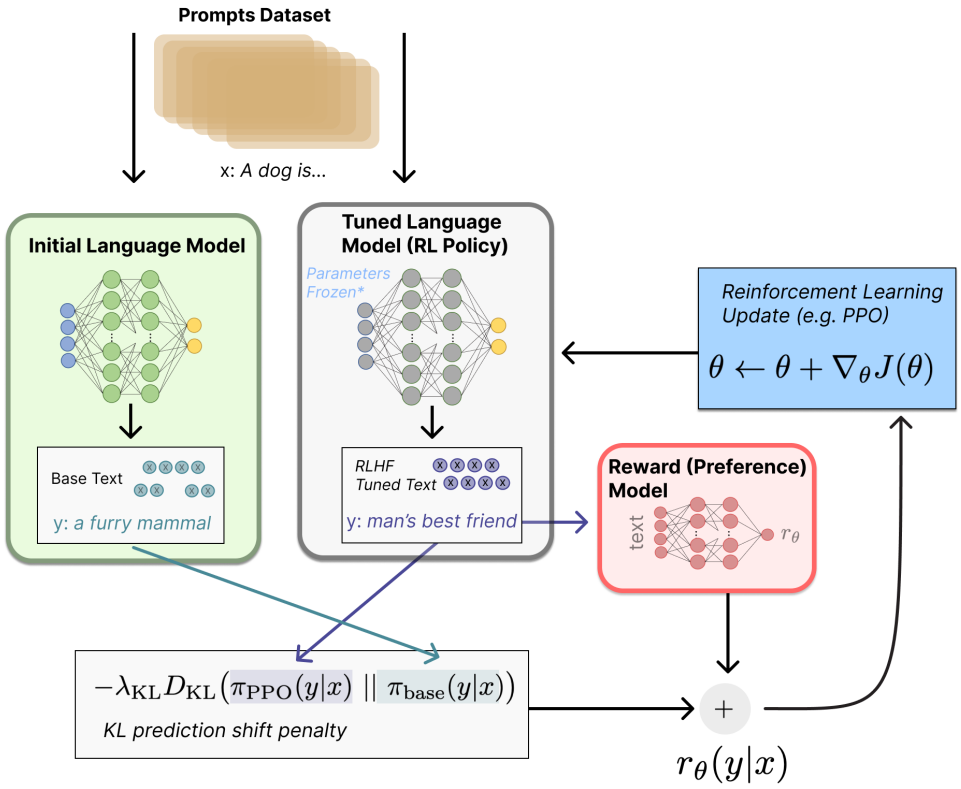


Figure 12 Detailed steps of Reinforcement Learning fine-tuning (HuggFace 2023)

As illustrated in 12, the active policy generates a response y using a given prompt x. The input x and response y are concatenated together and fed into the reward model to obtain a reward score r_θ. This score indicates the 'preferability' of the generated response y,

however model can exploit biases in the reward model leading to reward hacking. Meaning, if there is a bias toward specific keywords, number of tokens, or writing style, the model will generate those tokens repeatedly, resulting in an incoherent response (Ziegler et al. 2020), to address the reward hacking, Kullback–Leibler (KL) divergence is used. The KL divergence term penalizes the policy from substantially moving away from the initially pre-trained model, this ensures that the model outputs coherent text, without this, the model starts to generate an incoherent response and still receives a high reward. KL penalty is calculated by taking the difference of probability distribution of each token from an active policy with the base policy, $r_K L$. The final reward score used for updating the policy is $r = r_\theta - r_K L$.

Finally, the policy can be updated using methods like the Policy Gradient or Actor-critic method. However, Due to its stability and constraint on policy updates, Ziegler et al. (2020), Ouyang et al. (2022) and Ouyang et al. (2022) used PPO Schulman et al. (2017) method that maximize the reward score of the current batch of data.

RLHF process can be carried out by collecting preference data once and then proceeding with RL fine-tuning, or iteratively updating the reward model and policy together as demonstrated by Bai et al. (2022). They found that the reward model becomes less robust as the policy improves and gains higher reward, due to the lack of high-scoring data. To address this problem, they experimented with iterative 'online' RLHF, where they collected new comparison data using the best RLHF policy and retrained the reward model using this new data and subsequently training the new RLHF policy.

2.6 Low-Rank Adaptation

The advent of Transformer (Vaswani et al. 2023) gave raise to LLM such as GPT3 (Brown et al. 2020), BERT (Devlin et al. 2019), and LLAMA Touvron et al. (2023b). Although these models have shown exceptional performance on various NLP tasks, they come with huge computation overhead. The parameter size of these models ranges from a few million to hundreds of billions. Due to the size and complexity of these model, their practical use cases such as deployment and fine-tuning are restricted (Moez 2024).

To address this problem a parameter-efficient fine-tuning technique called Low-Rank Adaptation (LORA) was proposed by Hu et al. (2021). LORA significantly reduces the computation resources required by keeping the model's original parameter unchanged and simply adding a small feedforward layer to each corresponding original layer (Moez 2024). This technique leverages the concept of "low rank" from linear algebra, which allows for representing significant model changes with a minimal set of parameters. This

reduces the number of trainable parameters by 10000 times, thereby reducing the required computation resources and increasing the training speed.

The LLM are pre-trained with a huge corpus of unlabelled data sourced from the internet. As discussed in the previous section, the pre-trained LLMs are general-purpose models yet not optimized for specific tasks. The parameters of these models need to be fine-tuned to specialize for a particular task. However, due to the computational cost of this process, LORA presents a more efficient alternative. By adding a low-rank layer to each original layer of the model, LORA enables the fine-tuning of LLMs with significantly few trainable parameters, reducing both storage and computation requirements.(Moez 2024).

In summary, LORA presents a practical and efficient solution for customizing LLMs for specific tasks. This opens up new possibilities for the applications of LLMs in real-world applications.

3 Related Work

The field of RL has gathered more attention as researchers try to overcome its limitations, such as misspecified reward function. An important contribution in this area was made by Christiano et al. (2023) with their study "Deep Reinforcement Learning from Human Preference". Their work focuses on using human feedback to scale up reward models to enable deep reinforcement learning in complex, high-dimensional environments like Atari games (Bellemare et al. 2013) and physics simulator MujoCo (Todorov et al. 2012). This research proved that the human feedback mechanism could work in a complex setting that was previously considered too challenging, inspiring subsequent works that further investigate the use of human feedback in RL, especially text generation tasks (Ziegler et al. 2020; Ouyang et al. 2022).

Our research follows the work of Ziegler et al. (2020); Stiennon et al. (2022); Askell et al. (2021); Ouyang et al. (2022); Bai et al. (2022); Ramamurthy et al. (2023), similar to us, these studies are direct application of RLHF. They utilize human preference feedback or ranking as a reward to improve the quality of LLM for text generation in a domain such as summarization (Ziegler et al. 2020; Stiennon et al. 2022), translation (Kreutzer et al. 2018; Bahdanau et al. 2016), and dialogue generation (Askell et al. 2021; Ouyang et al. 2022; Bai et al. 2022; Thoppilan et al. 2022). For instance, both Ziegler et al. (2020); Stiennon et al. (2022) demonstrated that optimizing for human preference directly creates better summaries rather than optimizing for BLEU (Papineni et al. 2002) and ROGUE (Lin 2004). Furthermore, Stiennon et al. (2022), showcased that the reward model generalizes better to new domains without fine-tuning compared to supervised models.

Unlike some studies which focus solely on summarization or translation tasks, our work utilizes preference reward towards aligning general-purpose AI systems, similar to Askell et al. (2021); Ouyang et al. (2022). These later studies proved that fine-tuning LLM with human feedback is a promising direction for aligning it with human intention. Similarly, Bai et al. (2022) highlights the importance of this technique and shows how excessive focus on either $helpfulness$ or $harmless$ impacts the performance of another. They also demonstrated the advantage of iterative online training of preference model and RLHF policies on performance enhancement. Contrary to this, our work trains the reward model offline only once, emphasizing only $preference$ and assuming that annotators inherently favour the responses that are both $helpful$ and $harmless$.

Another form of human feedback explored in our research is *absolute* rating, currently employed in existing literature either during training (Kreutzer et al. 2018; Liu et al. 2018; Shi et al. 2021) or during inference (Freitag et al. 2022). For instance, similar to our work Kreutzer et al. (2018) used a Likert scale rating to improve the translation of product titles

on the e-commerce website and concluded that explicit rating doesn't improve the model, whereas implicit feedback does improve. They also found that both non-professional and expert annotators encountered challenges in consistently rating the translations of user-generated product titles. However, we found the validity of the rating dataset used to be questionable due to the potential influence associated with product images, the product itself and other factors unrelated to translation quality. These issues, combined with a lack of control over data quality were noted concerns. Whereas Liu et al. (2018), utilized simple explicit binary feedback to improve a task-oriented dialogue system.

Other research uses rating-based feedback (Liu et al. 2023a; White et al. 2023; El Asri et al. 2016) in the robotics and simulation domain. Noteworthy among these is White et al. (2023), which emphasizes that while preference feedback offers a practical approach it lacks the perspective of individual samples independently. Their work approaches absolute reward modelling as a multi-class problem within a traditional RL environment. This differs from our regression-based approach in the text generation context.

Our work differs from the existing research in two ways, one, we utilize an absolute reward for text generation as opposed to the work of simple translation (Kreutzer et al. 2018), or traditional robotic and simulation environment (Liu et al. 2023a; White et al. 2023; El Asri et al. 2016), or general-purpose text generation but with preference reward score (Ziegler et al. 2020; Stiennon et al. 2022; Askell et al. 2021; Bai et al. 2022). Second, we evaluate the effectiveness of combined reward signal achieve by integrating both absolute and preference feedback for text generation task. The goal is to aggregate the implicitly learned relative quality score from preference feedback and the absolute quality score learned from objective feedback over individual responses.

4 Methodology

This chapter describes the methodology of our research. It contextualizes the motivation driving our study and what objectives we aim to accomplish. Subsequently, it elaborates on the design of our experiments and how we demonstrate the outcome of our proposed approach. it also details the evaluation criteria to validate our hypothesis and finally concludes the chapter by discussing how we disseminate the outcome of the research.

4.1 Design Science Research Methodology

Our research adopts the Design Science Research Methodology (DSRM), as proposed by Peffers et al. (2007), to address identified limitation within RLHF pipeline used for fine-tuning LLM. The existing RLHF leverage preference reward model for RL step. This reward model implicitly learns to provide feedback from responses ranked relatively. The learnt feedback is relative to the quality of other responses and does not consider the objective quality of each response independently. Recognizing the potential drawback of this method, our study explores an absolute reward model, trained objectively on each response. The goal is to evaluate the efficacy of the absolute reward model compared to the preference reward model, this goal triggered the development of several RLHF models trained with both reward models.

DSRM is a research methodology used in Information System research which provides a systematic framework for designing and developing innovative artefacts to address specific problems or challenges in the field (Peffers et al. 2007). It encompasses several key steps: Problem Identification, Objective of a Solution, Design and Development, Demonstration, Evaluation and Communication. The subsequent sub-sections elaborate on how our research applies each step of DSRM framework to our study. A high-level overview of DSRM application specific to the Combine Reward Signal (CRS) RLHF study is depicted in Figure 13.

4.1.1 Problem identification and Motivation

As discussed in detail in the 'Introduction' chapter, the existing RLHF pipeline predominantly uses preference-based feedback for LLM fine-tuning. In this method, human annotators provide feedback by comparing different responses together and ranking them based on their preferences. The provided feedback is subject to other responses rather than considering the absolute quality objectively. Therefore there is a need to evaluate the effectiveness of absolute reward signal. We take the opportunity to study the impact of

incorporating absolute feedback, independently and alongside preference feedback in the RLHF pipeline and how it affects the quality of LLM generated responses.

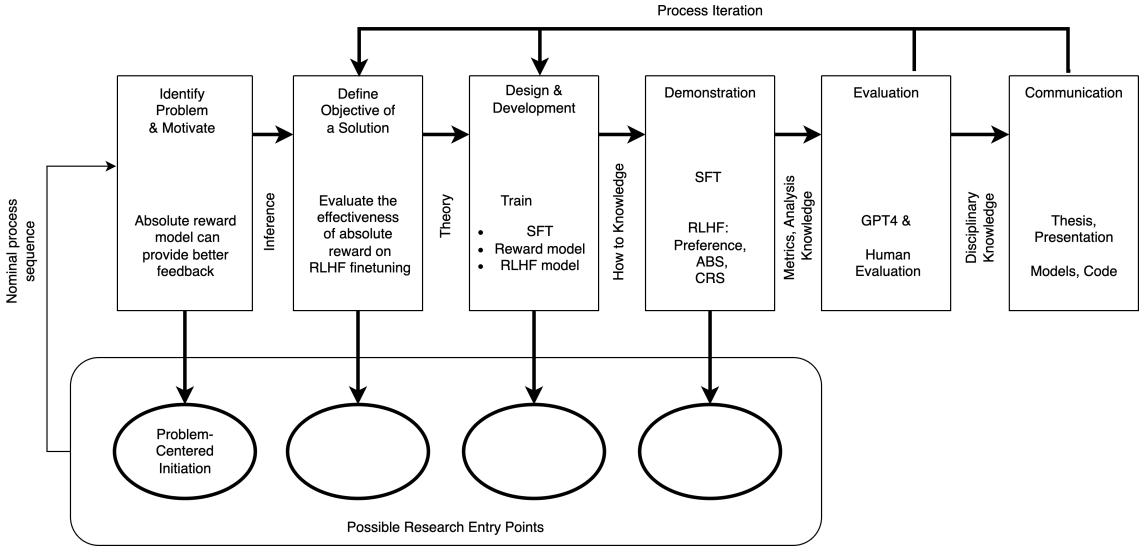


Figure 13 DSRM process for the CRS RLHF study (Peffers et al. 2007)

4.1.2 Objective of the Solution

The objective of this study is to investigate the nuance of using absolute reward signals and how it improves the training efficiency of this framework. The goal here is to answer the following questions that will help us fully understand how absolute reward signals work when used together with preference-based rewards.

How do preference and absolute reward modelling impact the performance and generalisability of RLHF models on various datasets?

The generated responses from each $rlhf$ model are ranked based on the received reward and the judgement of each reward model is compared with GPT4's judgement. The goal is to evaluate the generalisability of each reward model by comparing its judgement to the GPT4 judgement. The model which consistently agrees with GPT4 across various datasets is considered to be more roboust.

What is the effect of varying the relative weights of preference-based and absolute reward signals during the RL fine-tuning process?

The preference and absolute reward will be combined to generate a comprehensive reward signal by taking the weighted average. The weight of each signal is decided by experimenting with several values and observing its impact on the overall training process. The

question answers, how changing the weight of preference reward score in RLHF pipeline impacts the training efficiency.

What is the response quality and training efficiency of the RLHF model, when using only preference-based reward, only absolute reward or a combination of both?

The objective is to do a comparative analysis of RLHF models fine-tuned using various reward signals. In this stage, we use the absolute, preference, and combination of both rewards called Combined Reward Signal. Then compare the generated response from each $rlhf$ model in terms of coherency, informativeness, factual correctness and overall quality.

4.1.3 Design and Development

The development stage is divided into three steps and involves training a model which will be utilized in subsequent stages. Each model employs a different training data sourced from Open Assistant (OASST) (Köpf et al. 2023) dataset. This dataset consists of conversation trees with human-generated and annotated messages, offering high-quality preference feedback and absolute quality scores in different dimensions. Due to this reason, our research mainly uses the OASST to ensure a consistent training environment across all steps. The details of each step's training data, training procedure, hyper-parameter used, and evaluation criteria can be found in the 'Experiment Setup' chapter. The development stage consists of the following steps:

Supervised Fine-tuning: The first step of the pipeline is SFT. In this research, we use a LLAMA-based pre-trained model and fine-tune it on human demonstration from OASST Köpf et al. (2023) and other datasets. Through this fine-tuning process model learns to follow instructions. Which serves as a base for all subsequent models and will be used as a SFT baseline as well.

Reward Modelling: A reward model is needed to optimize the RL policy, which will be derived from two different types of feedback: preference-based and absolute feedback. To train the preference-based reward model, we refer to the work of Askell et al. (2021); Ouyang et al. (2022); Köpf et al. (2023). That implicitly trains a model to provide a scalar reward score indicating the quality of the response. The absolute reward model is trained either using regression or a binary classification algorithm. The choice of algorithm will be based on empirical analysis. The training detail of the preference model is explained in the 'Background' chapter, whereas details for the absolute reward model can be found in Section 5.3. Each reward model uses the previously fine-tuned SFT model as a base.

RL Fine-tuning: This step considers SFT model as an initial policy and fine-tunes it using the PPO (Schulman et al. 2017) algorithm. Where the policy generates a response for input prompts and optimizes itself based on the received rewards. It is the final step of RLHF pipeline where LLM models are fine-tuned to follow user intent and generate human-preferred responses.

4.1.4 Demonstration

The proposed solution will be demonstrated by fine-tuning several variants of the model. Each model will be trained by getting feedback either from preference, or absolute reward models. The models trained using this feedback are called abs_rlhf, and $preference\_rlhf$ respectively. An additional model called crs_rlhf is trained with CRS feedback by taking a weighted average of both preference and absolute reward scores. The weight for each signal is determined empirically and will be reported in the research.

4.1.5 Evaluation

The evaluation of all $rlhf$ models will be done by conducting a user study on the generated response from two proposed: abs_rlhf, crs_rlhf, and two baselines: $preference\_rlhf$, sft models. Since human feedback is expensive to acquire, our research primarily relies on automatic GPT4 annotators. Which evaluates each model by pairwise comparison and chooses the best one based on its capacity to produce informative and helpful responses. The model with the highest win rate will be considered the most preferable. This automatic evaluation is supplemented by human annotation but on a small dataset. Where each participant ranks responses from all $rlhf$ models according to their preference. The details of each evaluation method can be found in Section 5.5.

4.1.6 Communication

The research findings and results will be communicated through the thesis, presentation, and potentially open-source models uploaded on Hugginface[1] (Wolf et al. 2020). and code repositories[2]. The research aims to contribute to the field of RLHF by exploring the effectiveness of incorporating absolute reward signals in the training process.

[1] https://huggingface.co/alikhan0100u
[2] https://github.com/Ali1858/crs-rlhf

4.1.7 Contribution

The proposed research contributes to the existing body of knowledge by evaluating the impact of absolute reward signals in the RLHF pipeline. It compares the generalizability of the absolute reward model with the preference reward model and how it translates to RL fine-tuning of LLM. In addition to that, it also assesses the influence of combining preference-based and absolute reward signals. The research aims to enhance the quality of responses generated by RLHF models, thereby improving their overall performance in conversational tasks.

5 Experiment Setup

This chapter discusses the experiment procedures employed in the training phase of all three models within the Reinforcement Learning from Human Feedback (RLHF) pipeline. Following the constraints of our research each model utilizes a parameter-efficient fine-tuning technique called Low-Rank Adaptation (LORA) instead of full-finetuning unless it's specifically indicated. Additionally, the RL fine-tuning was done only for one epoch on a subset of data randomly sampled from the Open Assistant dataset (Köpf et al. 2023). This approach diverges from the conventional methodology which utilizes a large corpus of diverse data and trains for multiple epochs. Hence any conclusion drawn from these experiments should be interpreted with an understanding of these limitations.

The training process of SFT and reward model uses the HuggingFace library (Wolf et al. 2020) and follows the training procedure similar to the work of Köpf et al. (2023). For both models, our research substantially uses the code from Köpf et al. (2023) and repurposes it for our specific use case. Whereas RL fine-tuning utilizes the TRL library (von Werra et al. 2020) and follows their training examples as a guide.

The chapter systematically outlines the datasets used to train each model and describes its collection process. It also explains various decisions taken during the training process and defines the hyper-parameter selected for each model. Lastly, the chapter explains the evaluation criteria established to test our hypothesis, making it easier to understand the experiment outcomes.

5.1 Dataset

The RLHF pipeline consists of Supervised Finetuning (SFT), reward modelling, and RL fine-tuning, each step trained on different sets of datasets. This section describes the data collection process of *alpaca* (Taori et al. 2023), *dolly* (Conover et al. 2023), Grade School Mathematics (GSM8K) (Cobbe et al. 2021), and Open Assistant (OASST) (Köpf et al. 2023) datasets and how it is used in training each model.

5.1.1 Instruction Dataset

As described in Section 2.5, the pre-trained LLM has limitations and is not good at following human instructions. However, these limitations can be minimised with a SFT fine-tuning process. A supervised dataset also known as an instruction dataset is used for fine-tuning, where the dataset contains an optional input and instruction-response pair. The following open-source datasets are used for SFT step of RLHF.

Alpaca: LLM requires high-quality instruction (prompt) and response pair for fine-tuning. It is essential that instruction should be diverse and creative for the model to generalize well and follow unseen instructions. However, high-quality datasets are not easily available and are expensive to generate. Motivated by the fact, Wang et al. (2022) proposed an automated instruction generation process, known as SELF-INSTRUCT. Which uses LLM like GPT (Brown et al. 2020) to automatically generate diverse and creative instruction along with their input and output. It relies on the few-shot capability of the model to generate the dataset using a seed of 175 human written instructions along with their input and expected output. As shown in Figure 28, eight instructions are sampled from the pool out of which 6 are human-written and 2 are model-generated. Using these eight demonstrations (few-shot), the model generates new instructions. Subsequently, the model generates multiple instances of input-output pairs as depicted in Figure 29. After filtering the low-quality data, the generated instances are added to the pool. The data generation for *alpaca* (Taori et al. 2023) is based on the same approach, except instead of generating 2-3 instances, it only generates one instance per instruction. The *alpaca* dataset contains 52000 instances out of which 95% of data is used for training and 5% for evaluation.

Dolly: It is a human-written instruction dataset created by thousands of Databrick[3] employees. They were invited to create prompt-response pairs for eight different instruction categories. The dataset comprises of *instruction*, *context*, and *response* and is specifically designed to guide LLM to generate human-preferred responses. The *context* column is only necessary for categories like information extraction, closed Question and Answering (QA) and summarization, where annotators require reference text. The annotator will utilize the reference text and based on the task category either answer the question or summarize the reference text. It also contains instructions for tasks like classification, creative writing, open QA and brainstorming, where instruction should be answerable with general world knowledge without using the internet. The dataset contains 15000 instances out of which 95% of data is used for training and 5% for evaluation. In this research, we do not consider *context* information and treat all the categories equally.

GSM8K: Despite demonstrating impressive capabilities at several NLP tasks, LLM struggles at mathematical tasks which require multi-step reasoning (Cobbe et al. 2021). To address this GSM8K dataset was introduced by Cobbe et al. (2021). The dataset contains high-quality human-written grade school maths problem and solution pairs. To increase the interpretability the solutions are rather written in natural language instead of pure mathematical expression. The solution requires performing a sequence of two to eight steps with only a basic arithmetic operation to reach the final answer. The problems are

[3] https://www.databricks.com/

designed to be relatively unique making it stand out from other datasets where problems are either similar or vary only in minor aspects. The problems are moderately difficult for the state-of-the-art LLM and should be solvable by a bright middle school student. The dataset contains 8500 instances out of which 95% of data is used for training and 5% for evaluation.

Open Assistant: The Open Assitant is a high-quality multi-lingual conversational dataset crowdsourced from more than 13,000 volunteers. The data collection process is divided into 5 different stages; *prompting*, *labelling prompts*, *reply as prompter or assistant*, *labelling replies*, and *ranking assistant replies*. For SFT, labelling and ranking are not relevant and more information about these stages can be found in Sub Section 5.1.2. The conversational dataset not only demonstrates the instruction-following capability of LLM but also illustrates interactiveness via multi-turn dialogue between *prompter* and *assistant*. Making it suitable to fine-tune interactive and engaging chatbot assistants, similar to ChatGPT[4].

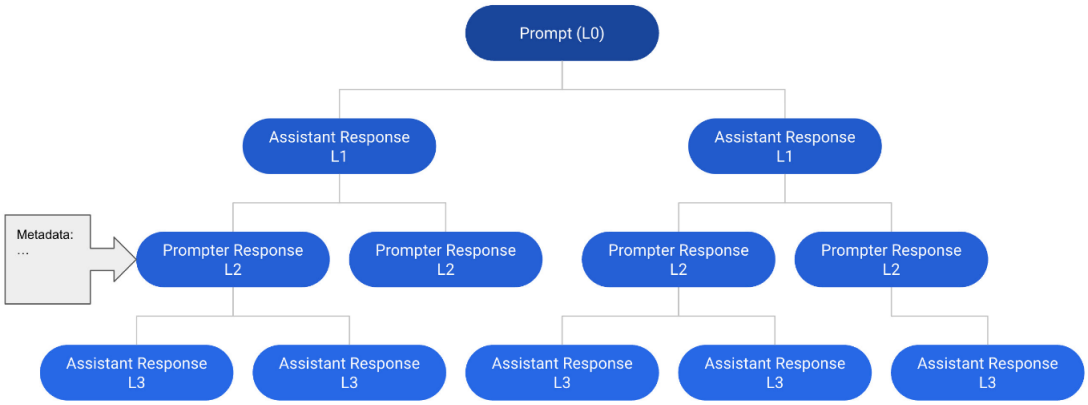


Figure 14 A sample Conversation Tree with a depth of 4 and 12 messages. Every path leading from the root prompt to any node represents a legitimate thread (Köpf et al. 2023)

The annotation process is divided into different steps and each step is performed by different annotators. It begins with *prompting*, where annotators contribute new prompts to a pool. These prompts are then selected via a lottery system to serve as the root node for a new Conversation Tree (CT). Then in the *reply as assistant* step, the annotator responds with a helpful and relevant answer after carefully considering the prompt. This step requires extensive research and requires a high-quality response from the annotator. Finally in the *reply as prompter* step annotators are encouraged to ask a wide array of questions to showcase various use cases, such as seeking clarification, changing the original query, or asking additional follow-up questions. These steps are repeated until it reach till certain

[4] https://openai.com/blog/chatgpt

depth or the conversation is completed. The quality of prompts and responses is ensured by quality control & content moderation. A designated *Trollboard* is maintained to pursue potential misbehaving annotators proactively. The rating of the board is determined by the aggregate of negative labels, reports, and downvotes received for their contribution.

The dataset consists of 66,497 CT with 161,443 messages in 35 languages out of which 10,968 CT are considered completed with their content moderation process finished. Among the incomplete trees, the majority of the trees only contain a single prompt. These incomplete trees will be utilised for RL fine-tuning which does not need the label. As illustrated in Figure 14 each prompt has multiple responses, which are ranked by annotators based on overall quality. The responses are sorted by their rank and threads with only the top one rank are chosen to ensure SFT is done on best demonstrations. Figure 16 shows the proposition of most common languages and as seen English and Spanish languages are dominated. Out of 35 languages, most languages are under-represented. Hence SFT only uses instances with the top 20 languages. This brings the dataset count to 9431 threads. The prompts and replies in the thread are concatenated together to represent a single input to SFT model. Out of 9431 instances, 95% of data is used for training and 5% for evaluation.

```
<|im_start|>user
{user prompt}<|im_end|>
<|im_start|>assistant
{Assistant answer}<|im_end|>
```

Figure 15 Prompt table for SFT fine-tuning and chatbots.(Köpf et al. 2023)

Prompt Template: The input for SFT model is prepared by concatenating the prompt and response together by using the template shown in Figure 15. Where $< |im\_start| >$ and $< |im\_end| >$ are tokens to indicate the beginning and end of a message by either *user* or *assitant*. The *user prompt* and *Assistant answer* are the placeholder for their messages. For a multi-turn conversation thread, the block is repeated till the conversation is over to concatenate the complete thread together, this serves as a single input to the model.

5.1.2 Reward Dataset

The preference and absolute reward model requires a supervised dataset with two different output labels. The preference-based model requires ranking labels and the absolute model needs a scalar score for training. Since the preference reward modelling approach is a commonly used reward score in RLHF pipeline, the training data is easily available, compared to the absolute scoring dataset. Despite the availability this research only uses OASST dataset because it provides a label to train both the reward models simultaneously,

as illustrated in Table 2. By utilizing the same dataset we ensure a fair comparison of the reward models. This section explains the ranking and labelling process for OASST dataset.

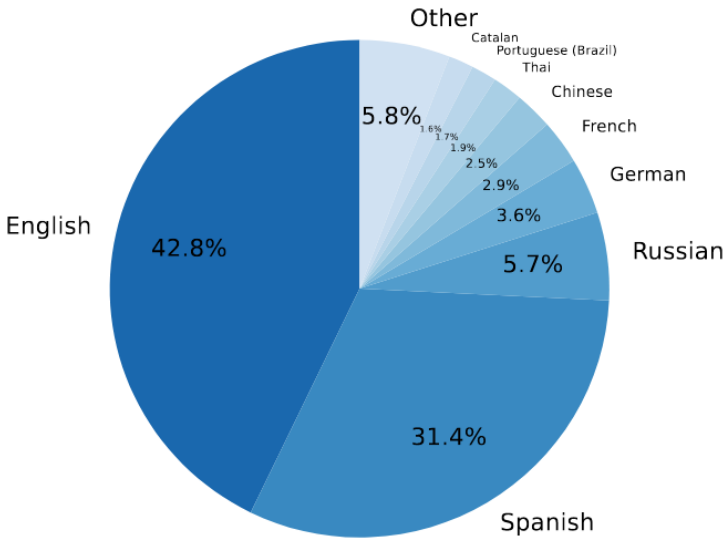


Figure 16 Proportion of the most common languages in the Open Assistant dataset.(Köpf et al. 2023)

In the *label a prompt* or *label a reply* stage annotators are presented with the messages along with previous conversation as a context. The annotators flag potential spam or troll messages and ensure they follow the guidelines. The language of the submitted message should be the same as the specified language and should not contain any personal identification, hate speech or, sexual content otherwise it is discarded by human moderators. Each message is rated on a five-point Likert scale indicating their creativity, humorousness, politeness, harmlessness, helpfulness and overall quality. Our research utilizes these ratings as a label to train the absolute reward model. Similarly, in the *rank assitant replies* stage, annotators are presented with two or more responses for a single prompt and its context. The annotators rank these replies according to their preference. To avoid the high variance of human preference, multiple ratings and rankings are collected for a single instance and later combined to form a consensus.

Similar to the instruction dataset, the proportion of the language in the reward dataset is also skewed toward a few dominating languages hence we only use the language which has enough representation to train the reward model such as English, Spanish, French and German. This brings the ranking dataset down to 14080 training and 1488 evaluation instances, where each instance is a combination of prompts and two to four responses. Since rating data treat each response independently, it contains 38852 instances for training data and 2,041 instances for evaluation. As illustrated in Figure 15 the messages are concate-

nated together and fed to the reward model as an input to predict the scalar score to either indicate the preference of the response with respect to another response or numerically show the *goodness* of the response independently.

Prompt	Responses	Rank	Labels
Why will open assistant be successful?	**Response 1:** Because I will be truly open and can be run on a desktop computer. So many people can run their own open assistant and tweak it towards their needs, be it personal, scientific or commercial ones.	1	**Violence:**0.0, **Creativity:**0.5, **Helpfulness:** 0.917, **Humor:** 0.167, **Toxicity:** 0.083, **Quality:** 0.833
	Response 2: Because I will be free. You'll not catch me saying, "As an AI language model." That is, unless you want me to.	2	**Violence:** 0.0, **Creativity:** 0.5, **Helpfulness:** 0.5, **Humor:** 0.75, **Toxicity:** 0.167, **Quality:** 0.5

Table 2 Sample from reward modeling dataset Open Assistant (Köpf et al. 2023)

5.1.3 Reinforcement Learning Dataset

Contrary to SFT and Reward model, the RL fine-tuning does not require the supervised dataset and only needs *user prompts* without *assistant answer*. However, it needs high-quality appropriate feedback. Since the reward model is solely trained on OASST dataset it makes sense to use the same for RL fine-tuning, ensuring the reward signal is from the same distribution. Similar to SFT and Reward dataset the conversation threads are concatenated together except, for this task, it ends with the prompt message instead of the assistant's response.

The RL fine-tuning process requires large-scale computation and due to time constraints and limited computation resources we made necessary adjustments in the training. After careful consideration, We decided to use partial data for hyperparameter tuning and training. The dataset contains 24000 instances and only 10% of that was used for hyperparameter tuning and 37.5% for final fine-tuning. This proportion of the data is used to maintain the balance between the minimum amount of necessary data and training time.

The subset was randomly sampled to be representative of the full dataset, ensuring that it captured the diversity and distribution of the original dataset as much as possible. The implications of the reduced dataset have been thoroughly considered and the result of the fine-tuning process is interpreted with knowledge of these constraints.

5.2 Training Supervised Fine-tuning

The SFT stage follows a similar training process as Causal Language Model (CLM). In this stage, all the corresponding tokens of assistant responses are masked and the model learns to predict them by generating the next tokens starting from $< |im\_start| > assistant$ till $< |im\_end| >$. We approached the instructing tuning process as a two-stage problem denoted by pre_sft and sft. The pre_sft stage initially fine-tune the model for one epoch with *alpaca* (Taori et al. 2023), *dolly* (Conover et al. 2023), GSM8K (Cobbe et al. 2021), allowing it to learn similar task. Later the sft stage refines the weight for two epochs to directly learn from OASST (Köpf et al. 2023) datasets. For both stages of training, the supported *max_token* is 2048 which includes both input-output sequences and the conversation thread larger than *max_token* is truncated. Table 3 shows the remaining hyper-parameter for both the stages.

Parameter	**Value**
Adam Beta 1	0.9
Adam Beta 2	0.999
Adam Epsilon	1e-12
Learning Rate (LR)	1e-4
LR Scheduler Type	Cosine
Warmup Ratio	0.03
Max Gradient Norm	0.3
Optimizer	Adam
Gradient Accumulation Steps	16
batch size	4
Lora r	16
Lora alpha	32
Lora Target Module	All

Table 3 Hyperparameters for pre_sft and sft fine-tuning process

This research uses the LLAMA (Touvron et al. 2023b) model for SFT task and after successfully fine-tuning, the model follows the instructions by generating relevant and coherent responses. However, during the RL fine-tuning, we observed that the training

process was not stable and the model diverged too much from the initial policy eventually leading it to generate an incoherent response. The potential reason for the deteriorating performance is discussed in the section 6.3

To address the unstable RL fine-tuning we utilize another LLAMA-based SFT model provided by Köpf et al. (2023), trained on OASST datasets for three epochs. This model serves as a base for reward modelling and RL fine-tuning

5.3 Reward Modeling

As discussed earlier, the reward function is a crucial component of RL system. However, in RLHF pipeline indirect human feedback is utilized through reward models to capture both implicit and explicit human intention. In this process, the reward model is trained to act as a proxy for human feedback, thereby eliminating the need for direct human involvement during the fine-tuning process. As depicted in Figure 11, the most common reward model is the preference reward model, in which the model is trained according to the annotator's preference. Table 2, sampled from OASST (Köpf et al. 2023), shows an example of a reward modelling dataset comprising a prompt, various responses, and their corresponding ranks and labels. The combination of *Prompt*, *Resonses*, and *Rank* constitutes a preference dataset, wherein annotators rank all the responses according to their preference. This study explores both an absolute reward model and a preference reward model. We employ the OASST dataset for the training of these models. One based on *rank* and the other based on *labels*. The dataset has the same input and differs solely at the output labels, allowing us to utilize the same data distribution for both models. This ensures the fair and direct comparison of the impact of different reward signals on RL model performance.

This research uses the LLAMA-based (Touvron et al. 2023b) SFT model as the base for both reward models. The final embedding layer for SFT model is replaced by a feedforward linear layer to output a single logit as a scalar feedback score. Both models use the same hyper-parameter configuration shown in Table 5 except for the batch size and are trained on OASST dataset for only one epoch to avoid over-fitting. The preference-base model uses a batch of 32 whereas the absolute reward model uses a batch of 128. The reward model supports *max_token* size of 2048, where the input prompt text can have a minimum size of 256 tokens and maximum size of *max_token* - *response_token_size*, if the input size exceeds the maximum value then it's truncated from the beginning of the text and remaining space is occupied by the response text.

While the specifics of preference-based reward modelling, including its loss function are detailed in Section 2.5.2, this section explains the training process of the absolute reward

model, covering aspects like experimenting with different weights for label aggregation, treating the reward model as logistic versus as a regression problem, and handling uneven distribution.

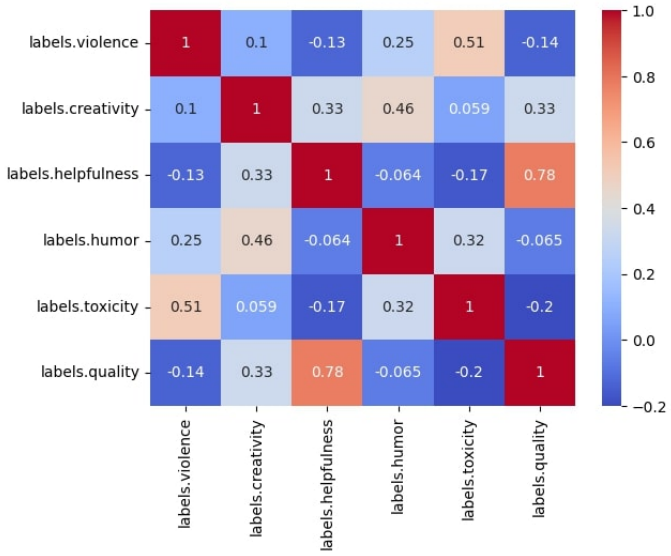


Figure 17 Open Assistant absolute reward labels correlation

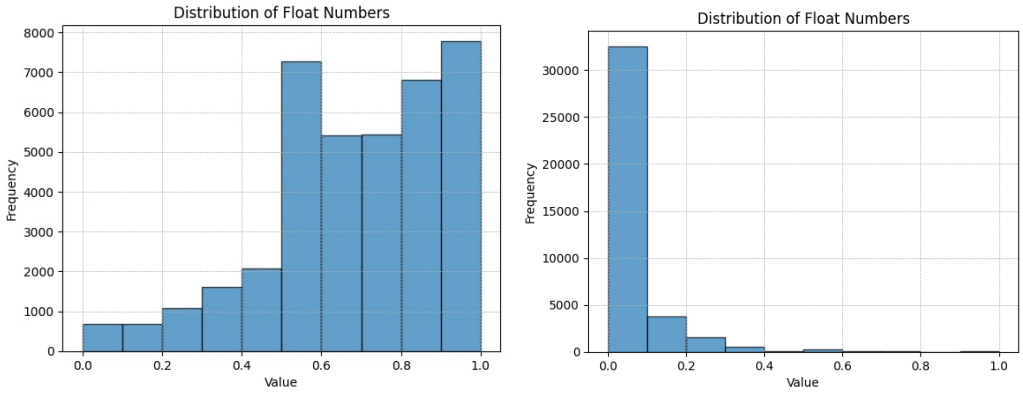
5.3.1 Exploring Reward Modeling Techniques: Aggregation, Under-Represented Data, and Task Type

Label Aggregation

The OASST (Köpf et al. 2023) dataset includes up to four different responses for a single prompt with their corresponding *labels*. As illustrated in Table 2, the dataset contains labels, indicating the absolute quality of each response. The training data for the absolute reward model is prepared by considering each response independently and creating a prompt-response pair, thereby increasing the training dataset up to 4 times. The reward model takes the prompt-response pair as input and outputs the scalar score. The scalar score for training data is created by aggregating all granular ratings into a single representative score. In this study, various weights for aggregation were explored, and two final weights were reported: $w2$ (denoted as $w2_{\text{wgt}}$) and $w3$ (denoted as $w3_{\text{wgt}}$) are given as follows:

- $w2_{\text{wgt}} = \{\text{violence} : -0.05, \text{creativity} : 0.15, \text{helpfulness} : 0.25,$
  $\text{humor} : 0.05, \text{toxicity} : -0.1, \text{quality} : 0.4\}$
- $w3_{\text{wgt}} = \{\text{violence} : 0, \text{creativity} : 0.15, \text{helpfulness} : 0.35,$
  $\text{humor} : 0, \text{toxicity} : -0.1, \text{quality} : 0.4\}$

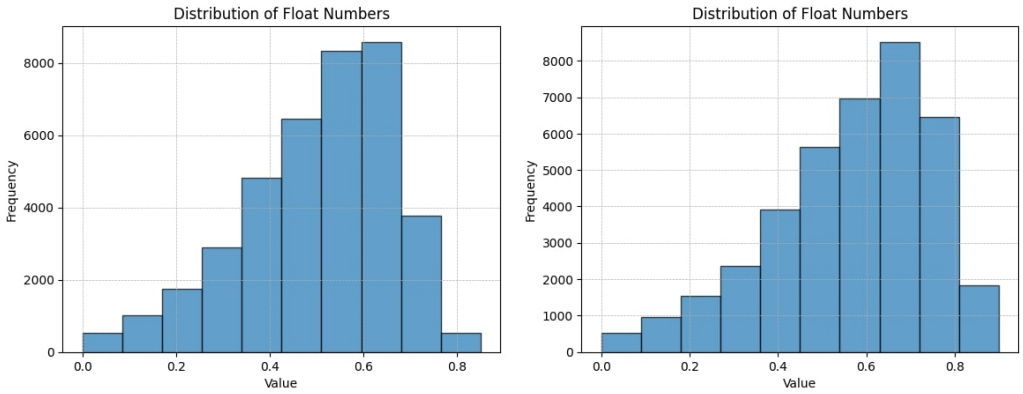
These weights are assigned according to our alignment goal. Since the *quality* label is a direct indicator of how good, informative and comprehensive the response is, it is assigned the highest weight. The $helpfulness$ has the second highest weight indicating that the generated response should be helpful. On the other hand, negative weights are assigned for both *violence* and *toxicity* due to their non-desirable traits. While *creativity* and *humor* are considered positive traits but not important, hence they are assigned low weights. Notably, the majority of the responses are non-violent as indicated by the skewed distribution toward a value of zero. This is illustrated in Figure 18b, leading to the assignment of zero weight for *violence* in $w3_{\text{wgt}}$. Similarly, due to the skewed distribution and incorrect labelling, *humor* are also assigned zero weights for $w3_{\text{wgt}}$ setting.



(a) Quality score **(b)** Violence score

Figure 18 Distribution of quality & violence label in Open Assistant dataset

However, It was observed that the reward model trained using both $w2_{\text{wgt}}$ and $w3_{\text{wgt}}$ configuration tends to give over conservative feedback, typically centred around the median of 0.5. The reason for such behaviour is evident when examining the distribution score in the training dataset. As depicted by Figure 19, there are very few instances of high-quality responses in the range of scores 0.8 to 1.0, with the majority being average. This leads to over-conservative feedback. To mitigate this issue, It was decided to only focus on the *quality* label to assess how good the generated responses are in terms of helpfulness and harmlessness. As demonstrated in Figure 17, there is a high positive correlation between *quality* and $helpfulness$, while *violence* and *toxicity* show a negative correlation. This makes the *quality* label an ideal candidate to represent the overall $goodness$ of the responses. Hence, from this point, all the experiments will be conducted with a focus on the *quality* label alone.



(a) W2 weight average **(b)** W3 weight average

Figure 19 Distribution of weighted average score

Handling Uneven Distribution

Utilizing the *quality* label in reward modelling presents the challenge of uneven distribution, as shown in Figure 18a. The values are heavily biased toward mid-quality responses and under-represent the high and low-quality responses. An analysis of the training data based on value range shows under-representation at both the high and low ends. There are 4104 instances in the low-end range (0 to 0.40), 7787 in the high-end range (0.9 to 1.0), and 26961 in the mid-range (0.41 to 0.89), out of a total of 38852 instances in the original training data, with an additional 2,041 instances in the evaluation set. To counter this imbalance, a combination of various approaches like over-sampling, under-sampling, data augmentation, and weighted loss are used. This section delves into these methods in detail.

The straightforward approach is to duplicate the under-represented data and reduce over-represented data, also known as over-sampling and under-sampling respectively. For low-end instances, data is duplicated up to 2 times (denoted as *over_s*) and for mid-range data, it is reduced up to 8000 instances (denoted as *under_s*), while keeping the data at the high-end range unchanged. To reduce the instance of mid-range data, a stratified under-sampling is done. The data in the range of 0.41 to 0.89 is divided into 5 bins and then equally sample 8000/5 instances from each bin, so that under-sampled data have equal representation.

An alternative to naive over-sampling is data augmentation (denoted as *augment*), which uses the paraphrased version of responses from under-represented data in addition to the original dataset. This approach involves pairing the prompt with paraphrased responses as input to the model. The SFT model is prompted to generate a 'clear, concise, and unnecessary details omitted' version from the original text, which often results in a low-quality

Algorithm 1 Algorithm for score adjustment based on text similarity

```
Import Sentence Transformer and Language Detection models
procedure ComputeSimilarities(GeneratedResponses, ReferenceResponses)
    for each pair of generated and reference response do
        Perform language detection
        Encode responses as text embeddings
        Calculate similarity using Cosine Similarity
    return list of similarity scores
procedure AdjustScores(OriginalScores, SimilarityScores)
    Define adjustment factor
    for each original score and corresponding similarity score do
        Normalize the similarity score
        Adjust the original score based on similarity
        Ensure adjusted score is within a valid range
    return list of adjusted scores
SimScores ← ComputeSimilarities(GeneratedResponses, ReferenceResponses)
AdjustedScores ← AdjustScores(OriginalScores, SimilarityScores)
```

response. Since these paraphrased responses might not represent the original responses, directly using the original feedback score would not be appropriate. Hence, as illustrated in Table 4 the augmented version of *label* is adjusted to represent the difference of the original feedback. The Algorithm 1 shows how adjustment of feedback scores is performed by evaluating the semantic and language similarity between the original and paraphrased responses by using Sentence Transformer (Reimers and Gurevych 2019, 2020) and language classifier (Conneau et al. 2020). It is essential to acknowledge that the quality of augmented data is expected to be noisier and of lower quality compared to original data, due to variations introduced by paraphrasing and adjuments in feedback scores.

Weighted loss is another approach for addressing under-represented data. This approach artificially inflates the loss from under-represented data through a multiplier *wgt*. This technique encourages the model to focus more on under-represented data. The *weight_value* is calculated by dividing the total size of the training data by the size of under-represented data. For example, the low-end over-sampled version of the data would have a loss weight of 5.23 (42956/8208). In our experiment, the full weight is not used, instead, we use weights at 50% and 75% strength, denoted as *wgt_s*_05 and *wgt_s*_075 respectively. If the constant is used it is denoted by *wgt_weight*.

Logistic vs. Regression Problem

In the context of the absolute reward model, which predicts continuous values, the task can be considered a simple regression problem. Since the range of the *quality* label is between 0.0 to 1.0, the predicted value should be confined within this range. However,

Paraphrased Version	Original Version	Original Score	Adjusted Score
"The Italian American said nothing to the IRS tax collectors."	"What did the Italian New Jerseyan say to the IRS tax collectors? Spaghett about it!"	0.333	0.20
"The dataset I was trained on primarily consisted of English material, which has resulted in my responses in Deutsch being less accurate. It is recommended to pose questions in English, as I comprehend it better."	"The dataset I was trained on was mainly in English. So my answers in Deutsch are not as good. You'd be better off asking me questions in English which I understand better."	0.25	0.17

Table 4 Samples from augmented dataset.

a simple regression model struggles to enforce this constraint. To handle this, several options are explored like utilizing a custom loss function, integrating a sigmoid layer, or redefining the task as binary classification. For the regression approach, a custom loss function is adopted, combining MSE and a penalty is used to constrain the predicted value. The penalty amount for two per cent of the value exceeding one or dropping below zero is added to the MSE loss, encouraging the model to avoid predicting such values.

The comparative analysis detailed in Table 6, reveals that the regression method with custom loss function (denoted as *mse + over_s + wgt_s_*075) is outperformed by both sigmoid-layer (denoted as *sig+ mse*) and logistic regression (denoted as *logistic*). The underperformance of *mse + over_s + wgt_s_*075 is due to its inability to predict values within the specified range, proving that manually constraining the prediction, as achieved via the sigmoid layer in the method *sig+ mse + over_s + under_s_*8*k* + *wgt_s_*075, is a better alternative. While this method employs a sigmoid layer to constrain the predicted value, it subsequently applies MSE for loss calculation. Conversely, *logistic + over_s + wgt_s_*075 utilizes the sigmoid layer but opts for BCE as its loss function. The two different version of models, *augment + under_s_*8*k* and *augment + over_s_*8*k + under_s_*16*k* + *wgt_*1_4, trained with the same configuration except for different loss functions perform equally average as long as the output of the model is constrained by using a sigmoid layer. The decision of choosing between *logistic* and *mse* is discussed in the next section.

However, it is worth noting that there is potential for further optimizing the *mse + over_s + wgt_s_*075 approach by experimenting with the different penalty values. Currently, we proceed with the task using the sigmoid layer to predict the rewards within the [0,1] range.

5.3.2 Absolute Reward Models Comparative Analysis

This section does a comparative analysis to aid in the selection of the final reward model based on its performance on the evaluation dataset. As evident in Table 7 all the models perform well for mid-range scores and either struggle at high-end or low-end. The goal is to find the model which balances the performance across all ranges of scores. Based on the evaluation metrics in Table 6 both *logistic* and *sig + mse* model struggle at the tail, with the *logistic* model performing well at the low-end range and the *mse* model at the high-end range. In RL fine-tuning it's important to provide appropriate feedback, especially for extreme action. This helps the agent to modify its action accordingly. Hence, we prioritise the method which performs equally well at the low end and balances it at the high end. Based on this fact and metrics in Table 6 we decided to use logistic-based models because they provide proper balance between extreme ends. Ideally, the reward model should perform well for all ranges of scores, however, due to uneven distribution and noise in the dataset, performance is low for *very bad* and *very good* responses. Hence, the model which performs equally well for extreme responses will be chosen.

Parameter	**Value**
Adam Beta 1	0.9
Adam Beta 2	0.95
Adam Epsilon	(not specified)
LR	5e-5
LR Scheduler Type	Cosine
Warmup Ratio	0.03
Max Gradient Norm	0.3
Optimizer	Adam
Gradient Accumulation Steps	16
Lora r	16
Lora alpha	32
Lora Target Module	All

Table 5 Hyperparameters for absolute reward model training

To handle the under-represent value at the low end, several experiments are conducted on a combination of over-sampling (*over_s*), augmentation (*augment*) and weighted loss

method	Below 0.4		Above 0.9		Between 0.4 and 0.9	
	mse	mae	mse	mae	mse	mae
logistic + over_s + wgt_s_075	0.0771	0.2348	0.0862	0.2765	0.0293	0.139200
mse + over_s + wgt_s_075	0.3159	0.5161	0.06420	0.1971	0.1209	0.2801
sig + mse + over_s + under_s_8k + wgt_s_75	0.0618	0.1981	0.0964	0.2800	0.0423	0.1652
logistic + over_s + under_s_8k + wgt_s_075	0.0611	0.1999	0.1027	0.2923	0.0415	0.1662
logistic + augment + under_s_8k	0.0853	0.2446	0.0786	0.2493	0.0326	0.1455
sig + mse + augment + under_s_8k	0.1056	0.2827	0.0593	0.2132	0.0299	0.1360
sig + mse + augment + over_s + under_s_16k + wgt_1_4	0.1223	0.3030	0.0441	0.1796	0.0255	0.1262
logistic + augment + over_s + under_s_16k + wgt_1_4	0.0889	0.2555	0.0645	0.2320	0.0263	0.1301

Table 6 Comparative analysis for absolute reward model: Regression vs Logistic

(*wgt_s_*05 and *wgt_s_*075) along with the under-sampling of mid-range values to 8000 (*under_s_*8*k*) and 16000 (*under_s_*16*k*) instances. Analysis of the result reveals that the method with *augment* and *under_till_*8*k* with weight *wgt_s_*075 gives balance performance for both high-end and low-end scores, for instance, *augment* +*under_s_*8*k* + *wgt_s_*075 and *logitstic* + *augment* +*under_s_*8*k* as illustrated in Table 7 and 6 respectively. After carefully considering the evaluation metrics we identified four models and manually evaluated the feedback score from these models

We use ChatGPT to create several prompts and multiple responses for each prompt as shown in Table 24,25,26, and 27. These responses are specifically designed to evaluate the reward model, where response 1 is the *best* answer and response 2 is the *bad* one. Two other responses are added to test the model's feedback score on irrelevant and meaningless answers, where response 3 is *irrelevant* and response 4 is *meaningless* one. Ideally, the reward model should provide the feedback score in such a way that it ranks responses similar to the given order. Based on the score it is clear that the model *logistic* + *over_s* + *wgt_s_*05 is not able to differentiate between *bad*, *irrelevant*, and *meaningless* responses, while the *augment* based model performs better. We observe that among all the *augment*-

method	Below 0.4		Above 0.9		Between 0.4 and 0.9	
	mse	mae	mse	mae	mse	mae
augment + under_s_8k + wgt_s_05	0.1052	0.2799	0.0568	0.2080	0.0275	0.1322
augment + under_s_8k + wgt_s_075	0.0721	0.2267	0.0852	0.2640	0.0335	0.1471
over_s + under_s_8k + wgt_s_05	0.0519	0.1801	0.1195	0.3174	0.0465	0.1754
over_s + under_s_8k + wgt_s_075	0.0611	0.1999	0.1027	0.2923	0.0415	0.1662
over_s + under_s_8k	0.1051	0.2752	0.0534	0.2009	0.0290	0.1351
over_s + wgt_s_05	0.0771	0.2348	0.0862	0.2765	0.0293	0.1392
over_s + wgt_s_075	0.0591	0.2040	0.1159	0.3205	0.0369	0.1573
over_s	0.1227	0.3099	0.0518	0.2091	0.0222	0.1192

Table 7 Comparative analysis for absolute reward model: logistic regression with uneven distribution

based models, the *logistic + augment + under_s_8k + wgt_s_*075 is the clear winner and generally it can differentiate between *bad* and *irrelevant* answers. Hence, we choose this as the *best* absolute reward model for RL fine-tuning.

It is important to note that this analysis is based on the ranking because it is not always clear if the assigned reward score makes sense especially when two responses are very similar in quality. This proves that comparing and ranking the responses is easier than assigning an absolute score, as already stated by Christiano et al. (2023). This can lead to inconsistent and noise annotation compared to the preference-based reward model. The section 8.2 discusses this issue further.

5.4 RL Fine-tuning

The RL fine-tuning is the final step of the RLHF pipeline. This stage fine-tunes the SFT model using a RL algorithm and a reward model. In our research, we fine-tune the LLAMA-based (Touvron et al. 2023a) SFT model using PPO algorithm and used several different reward models to provide reward signal for fine-tuning. We experimented with different hyper-parameters and found the hyper-parameter mentioned in Table 8 generating coherent and meaningful responses. The model supports the input of size 1024 tokens and output of 512 tokens, bringing the total *max_token* size to 1536 tokens. To reduce the memory requirement the decision was made to discard the prompt with a larger size than 1024 tokens, even though SFT supports *max_token* 2048, which is the basis for RL models. Hence, all the RLHF experiments follow this configuration to maintain consistency across different experiments, unless specifically indicated. This section describes the details of different reward models used for RL fine-tuning.

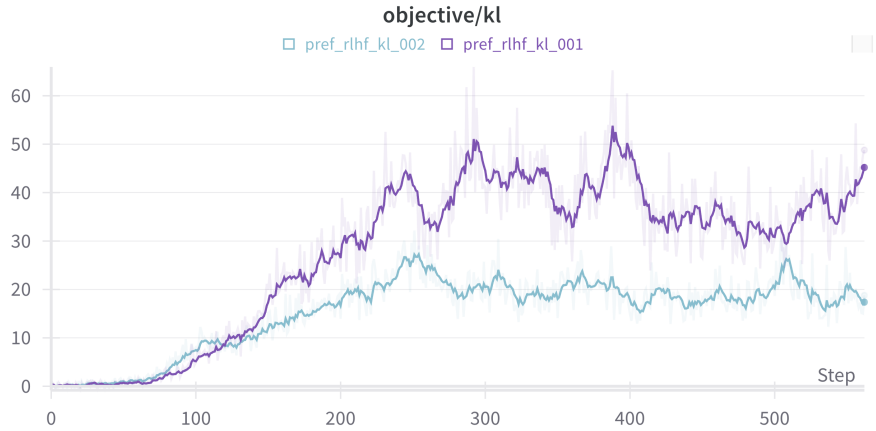


Figure 20 Preference RLHF: Initial KL co-efficient 0.01 vs 0.02

Preference Reward. Following the work of Ouyang et al. (2022); Askell et al. (2021); Bai et al. (2022) we used a reward model trained on the preference dataset. As seen in previous sections, this model learns by comparing the generated responses and implicitly provides a scalar score for each prompt-response pair, the RLHF model trained with the reward signal is denoted by $preference\_rlhf$ and servers as a RLHF baseline. We experiment with an Initial KL co-efficient of 0.02 ($pref\_rlhf\_kl\_002$) and 0.01 ($pref\_rlhf\_kl\_001$) and found that the model $pref\_rlhf\_kl\_001$ performs better compared to $pref\_rlhf\_kl\_002$ model. However, as seen in Figure 20 the KL objective for the $pref\_rlhf\_kl\_001$ model is unusually high and isn't considered to be good (von Werra and Belkada 2023). The high KL value means the model is diverging too far away from an initial policy. Since the KL value for the $pref\_rlhf\_kl\_001$ model is high and shows an upward trend, this can eventually lead to generating an incoherent response while training with the complete dataset. Hence we use Initial KL co-efficient of 0.002 for all the subsequent experiments.

Absolute Reward. Contrary to $preference\_rlhf$ the second model utilizes the absolute reward model to obtain a reward signal and is denoted as abs_rlhf. As explained in the reward modelling section, the absolute reward model uses a quality score ranging from zero to one. To achieve this sigmoid layer is used for constraining the predicted values within this range. We trained the abs_rlhf model for more than 300 steps with two different settings of the absolute reward model. One with a sigmoid layer (denoted as $abs\_rlhf\_bounded$) to constrain the predicted reward and another without the sigmoid layer (denoted as abs_rlhf_logits) and found that the $abs\_rlhf\_bounded$ model generated smaller responses as training progress. As evident in Figure 21 the mean response length is decreased in comparison to abs_rlhf_logits. This is also demonstrated in Table 28 with an example.

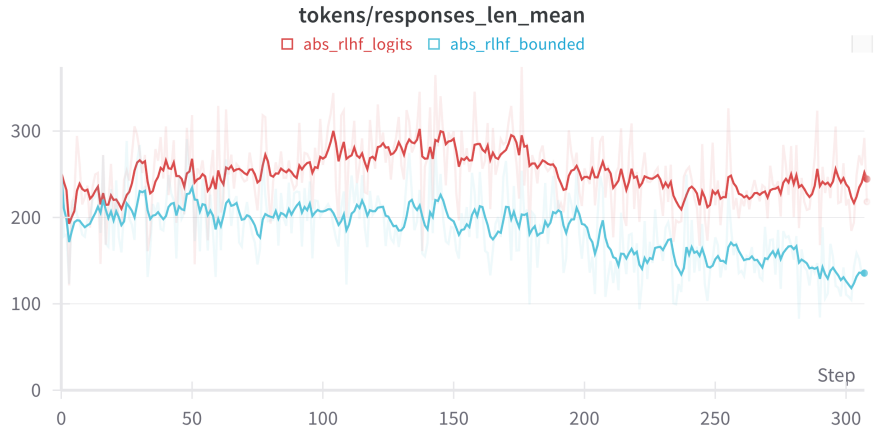


Figure 21 Abs RLHF: Reward as Logits vs Probability

While investigating we found the potential reason for this behaviour is that using the logits provides nuance feedback compared to *abs_rlhf_bounded*. Considering the score from *logistic+augment+under_s_8k+wgt_s_*075 reward model in Table 27 as an example, we have the reward of 0.6914, 0.4258, 0.3496, 0.3398 for all four responses. Converting these scores back to logits we get 0.8067, -0.2990, -0.6208, and -0.6642 respectively. Using the logits as a reward instead of a bounded score provides the RL model with a stronger positive and negative reward signal as shown by the difference in score for responses 1, 2, and 3. This is especially useful when two responses are of similar quality, for example, response 3 and response 4 with their respective score of 0.3496 and 0.3398. Hence the final *abs_rlhf* and subsequent *crs_rlhf* fine-tuning will be performed with logits as a reward.

In future research, It would be intriguing to explore the outcome of *abs_rlhf* when using the absolute reward model trained directly with logits as a reward signal instead of a bounded score. However, for this research we proceed with the same model without a sigmoid layer.

Combined Reward. The Third model aims to leverage both implicit and explicit feedback hence it employs a combination of both preference and absolute reward signals for fine-tuning. This model, denoted as *crs_rlhf*, calculates a weighted average of these two reward scores, where the weights are normalized and range from zero to one. The preference reward weight is determined by empirical analysis, while the weight of the absolute reward model is set as the complement of the preference reward weight (1 - preference reward weight). We conducted experiments with five different weight proportions: 0.25, 0.375, 0.5, 0.625 and 0.75, training the *crs_rlhf* model on 10% data for each weight

scenario. The result of these experiments is detailed in the next section and based on the result we choose the optimal weight to fine-tune the final crs_rlhf model.

Parameter	**Value**
Adam Beta 1 & Beta 2	0.9, 0.95
Adam Epsilon & Weight Decay	1e-8, 1.0e-6
LR	1.41e-5
Batch size	16
PPO epoch	4
Initial KL co-efficient	0.002
Clip range & value	0.4
Score clip	4
Lora r	16
Lora alpha	32
Lora Target Module	All

Table 8 Hyperparameters for Reinforcement Learning fine-tuning

5.5 RL Evaluation

The evaluation set of RLHF is curated by sampling the user prompts from Vicuna (Chiang et al. 2023), Koala (Geng et al. 2023), HH-RLHF (Bai et al. 2022), and the held-out version of OASST (Köpf et al. 2023) dataset. Vicuna dataset is derived from the user-shared conversation upload on ShareGPT[5], a website to posts the user's conversation with ChatGPT. It is a clean version of the dataset obtained by removing inappropriate and low-quality samples. Similarly, the Koala test set is also sampled from the publicly available user-written prompt. In addition to that, we repurposed the HH-RLHF dataset for RL evaluation, which is a helpfulness and harmlessness dataset for preference model training. The curation step of the eval dataset ensures that the prompts only contain a single message and that the model has been not already trained on these user prompts. This decision to use a single-user prompt is taken to keep the evaluation process simple during user study and to keep the cost of automatic evaluation as low as possible. Since the training dataset for both RL fine-tuning and reward modelling does contain the multi-turn conversation threads between $user$ and $assistant$, we assume the evaluation result from a single prompt should be replicable to multi-turn conversion.

Additionally, our research doesn't explicitly evaluate the $rlhf$ models for $helpfulness$, $harmlessness$ and assumes that in general, the preferred human responses are implicitly

[5] https://sharegpt.com/

based on this factor, as it is already stated in the guideline. Hence the evaluation process only focuses on human preference. Furthermore, this section describes the evaluation process in detail and how we leverage the dataset to evaluate the $rlhf$ models.

5.5.1 Automated Evaluation

The emergence of the LLM and advancement in deep learning-based technique has made significant progress in the field of NLP, especially for generative tasks like machine translation, summarization, story generation, dialogue generation, QA, and many others. However, effectively evaluating the model on generative tasks is a challenging problem. The traditional automatic evaluators like BLEU (Papineni et al. 2002) and ROGUE (Lin 2004) are used to evaluate translation and summarization tasks respectively. These evaluators focus on the number of overlapping words between the reference and generated text while ignoring the semantic nuance between them, leading to low scores even if the generated text is semantically similar to the reference text but differs in phrasing. This makes the approach unsuitable to evaluate RLHF models, as it doesn't align with human preference (Ziegler et al. 2020; Stiennon et al. 2022). Other pre-trained LLM based evaluator such as BERTScore (Zhang et al. 2023), MoverScore (Zhao et al. 2019), BARTScore (Yuan et al. 2021), BLEURT (Sellam et al. 2020) use semantic similarity to evaluated the generative model. Even though these methods are better than traditional ones, they are still far from perfect and also require reference text (Gao et al. 2023), which are often expensive to acquire.

Recent studies propose to use LLM such as GPT-3 (Brown et al. 2020) for automated evaluations that mimic human judgment (Gao et al. 2023; Fu et al. 2023; Liu et al. 2023b; Luo et al. 2023; Wang et al. 2023; Zheng et al. 2023). The LLM have the ability to understand the task and respond from within the context provided without needing any additional training. This makes them a cheaper alternative to human evaluators for evaluating several complex NLP tasks. Luo et al. (2023) demonstrates the ability of ChatGPT to effectively evaluate the factual inconsistency in summarization tasks while highlighting its limited reasoning ability. Whereas, Gao et al. (2023) compared the capability of ChatGPT with human evaluation such as Likert scale scoring and pairwise comparison for evaluating summarization tasks and found it performs better or the same as human evaluators. Fu et al. (2023) proposed a LLM based framework called GPTScore, a customizable and wide-ranging evaluation approach that covers 22 evaluation aspects across 37 datasets. Liu et al. (2023b) propose G-Eval, a framework based on Chain-of-Thoughts (CoT) (Wei et al. 2023) to use GPT-4[6] as a automated evaluator for text summarization and dialogue generation task. Zheng et al. (2023) uses $LLMs\ as\ judges$ to evaluate chat

[6] https://openai.com/gpt-4

assistants on open-ended questions and achieved 80% agreement with human preferences. Although these evaluators are biased and have limited reasoning ability they are proven to be highly correlated and cheaper alternatives to human evaluators and also do not require reference text for evaluation. Hence, Following their work, our research utilizes GPT-4 Turbo[7] for automatic pairwise comparison between candidate models.

Our research uses the AlpacaEval (Li et al. 2023) library for automatic evaluation. The library provides an option of choosing from several popular closed-source LLM as an annotator, including GPT-4 and GPT-4 Turbo. The meta-evaluation of different annotator done on the AlpacaEval dataset with 650 instructions and 4 human annotations each reveals GPT-4 (denoted as $alpaca\_eval\_gpt4$) annotator has highest human agreement of 69.5 and GPT-4 Turbo (denoted as $alpaca\_eval\_gpt4\_turbo\_fn$) being third with 68.1. Considering the cost limitation we decided to use $alpaca\_eval\_gpt4\_turbo\_fn$ annotator which is significantly less expensive than $alpaca\_eval\_gpt4$. It has a Spearman correlation of 0.93 with human annotation and has a probability of 0.65 that it prefers a longer response out of two responses if the difference between their length is significant.

The annotators simply use the template illustrated in Figure 31 and prompt LLMs to act as evaluators. The annotators provide two metrics: Win Rate and Standard Error. The win rate is calculated by taking an average of the number of times model responses are preferred over reference responses. For each instruction, the model response is paired with the reference response and sent to the annotator for ranking. The Standard Error (S.E) of the win rate that measures its variability. In our case comparison is done with another fine-tuned model, Hence we pair the model response with another model's response instead of a reference. It also randomizes the sequence of model responses to reduce the position bias of the annotators.

AlpacaEval (Li et al. 2023) library performs paired t-test to find the minimum number of required samples to tell each model apart. The analysis was done among 78 models using $alpaca\_eval\_gpt4\_turbo\_fn$ annotator on an eval dataset of 805 samples and found the majority of model pairs are distinguishable with 150 samples. Considering this recommendation and our cost limitation, we decide to use 100 prompts to evaluate all RLHF models. Our evaluation set is created by a stratified sampling of 100 prompts from the Vicuna (Chiang et al. 2023), Koala (Geng et al. 2023), HH-RLHF (Bai et al. 2022), and OASST (Köpf et al. 2023) dataset. The dataset is known as the $final\_eval$ dataset. Along with this, another set of 50 prompts, known as $tuning\_eval$ are sampled from OASST dataset for hyper-parameter tuning and debugging. The sampling process ensures that there is no overlap between evaluation sets, making the evaluation results more reliable.

[7] https://help.openai.com/en/articles/8555510-gpt-4-turbo

The *tuning_eval* dataset is used to find the optimal weight of crs_rlhf model by comparing crs_rlhf_025, crs_rlhf_0375, crs_rlhf_05, crs_rlhf_0625 and crs_rlhf_075. The final evaluation is done to compare the $preference\_rlhf$, abs_rlhf, crs_rlhf (trained with best weight) and sft on the $final\_eval$ dataset.

5.5.2 Human Evaluation

The human evaluation is conducted to evaluate all *rlhf* models and serves as a supplementary evaluation in addition to the automated evaluation. This stage finds the correlation between GPT4 and humans for our specific use case and provides us confidence in the automation evaluation step.

Similar to the $final\_eval$ dataset, the *human_eval* is also prepared by stratified sampling the Vicuna, Koala, HH-RLHF, and OASST dataset. However, human feedback is difficult and expensive to acquire and due to this fact we only sample 20 prompts instead of 100. The sampling process starts with a random sampling of almost 25 prompts from each dataset and later manually handpicked only 5 prompts from each subset, ensuring no overlap with $final\_eval$ and *tuning_eval* set. The manual picking is done after observing the wide topics, from physics to cooking, to coding covered by the subset. Hence the careful decision was made to only conduct human evaluation on the general topic which doesn't require special knowledge expertise. It should be noted that manual hand-picking and using only 20 prompts introduces biases. The decision to select 20 prompts for only general topics was made due to the limitation of human annotators and the result from the human evaluation should be interpreted with this understanding.

The LLM generate a sequence of words iteratively and this generation process is designed to mimic humans and by nature, the human-written text is not the most probable (Holtzman et al. 2020). Therefore, language generation is stochastic in nature in order to generate favourable and non-repeating text. They often use techniques like top-k (Fan et al. 2018) and top-p (Holtzman et al. 2020) sampling to decide the next word in a sequence. It randomly samples the next token from top-k or top-p most probable tokens. Since the generation process involves randomness, different runs from the same prompt can lead to generating different quality of the text, which subsequently impacts the evaluation result. Hence to minimize this variability, the evaluation should be conducted on a large enough sample size. However, due to the limitation of human annotators, it is not possible in our research. Therefore, To mitigate the issue of stochastic language generation, we run three instances of language generation for each prompt, creating three different sets of responses for the same prompts. Considering this aspect, the human evaluation is conducted with 12 graduate-level participants divided into three equal groups of annotators,

where each group annotate the same 20 prompts but with different responses. The final evaluation result is computed by taking an average from all groups.

The evaluation is conducted using an open-source software called Argilla (Vila-Suero and Aranda 2023). It provides the functionality to create an interactive user interface for data collection, making it easier for humans to annotate the response. An example of annotation data is illustrated in Figure 30. On the left side, it showcases user prompts along with its three responses, whereas on the right side, it provides annotation labels. The label section contains three button elements named 'Assitant Response 1', 'Assitant Response 2' and 'Assitant Response 3' representing the model responses. The annotator's task is to select the button and drop it into the box corresponding to the first, second or third rank according to their preference. It also provides the option to assign the same rank for different responses in case of a draw.

Annotator	**Ranking**	**Tideman Merged Ranking**
Annotator 1	response-1 → rank 2 response-2 → rank 1 response-3 → rank 3	response-1 → rank 2 response-2 → rank 3 response-3 → rank 1
Annotator 2	response-1 → rank 2 response-2 → rank 3 response-3 → rank 1	
Annotator 3	response-1 → rank 2 response-2 → rank 3 response-3 → rank 1	

Table 9 Annotator Rankings for a prompt and Tideman Merged Ranking

Each group consist of 4 annotators evaluating 20 prompts each with three responses, corresponding to the answer generated from $preference\_rlhf$, abs_rlhf and crs_rlhf models in randomized order to reduce the positional bias. Since automated evaluation already compares all the $rlhf$ models to the sft baseline on $final\_eval$ dataset, the human evaluation is only done among $rlhf$ models to reduce the workload of the human annotators and increase the reliability of the evaluation result. Due to multiple annotations on the same prompts, each response rank has overlapping votes which need to be merged. Our research utilizes Köpf et al. (2023) implementation of "Tideman's method" (Tideman 1987) to merge ranked-votings. This voting method is a way of deciding an election between multiple candidates using voters' ranking of those candidates. In our case, it provides a fair way of determining a winner's responses for each prompt when multiple annotators (voters) rank candidate responses in order of preference. An example of the Tideman vot-

ing method is illustrated in Table 9. It showcases the ranked annotation of a prompt along with its merged Tideman voting. Once we have merged the ranking for all 20 prompts, it can be used to calculate the win rate of each model.

Contrary to the pairwise comparison between models in automated evaluation, the human evaluation task collects ranking feedback, where the annotator ranks three responses according to preference. The ranking feedback doesn't allow direct calculation of the absolute win rate, which is possible with pairwise comparison. Instead, we aggregate the rank of each model over all 20 prompts with a simple rank aggregation method, called Borda count (Emerson 2013). Our research uses a normalized version of this aggregation method and assigns points 1 for the first rank, 0.67 for the second rank, and 0.33 for the third rank. The points of each model are calculated and averaged over all prompts and sorted in descending order. Where the model with the highest point is the best and with the lowest point is the worst.

This process of merging each model rank from different annotators and aggregating their ranks for all prompts is repeated and averaged over all annotation groups to estimate the mean rank for each model.

6 Result

This chapter presents the findings of our research. It showcases a detailed analysis of our Reinforcement Learning from Human Feedback (RLHF) fine-tuned model supported by statistical evidence. It starts with the hyper-parameter tuning to find the optimal weight of preference reward score in Combine Reward Signal (CRS) model and investigates the trade-off between both reward models while aggregating them. Then illustrate the comparison result between preference and absolute reward models. It also demonstrates how double LORA doesn't work in our experiment and finally shows how each RLHF model is performing according to GPT4 and human annotators.

6.1 Trade off between preference and absolute reward model

Finding Optimal CRS Weight: The aggregation of the preference reward score and absolute reward score is done by taking a weighted average, where the weight of the preference reward score is chosen empirically after experimenting with several combinations of weights, whereas the weight of the absolute reward score is complementary. Experiments were conducted with 10% of training data with different weights such as 0.25, 0.375, 0.5, 0.625, 0.75 denoted by crs_rlhf_025, crs_rlhf_0375, crs_rlhf_05, crs_rlhf_0625, crs_rlhf_075 respectively, and evaluated on *tuning_eval* dataset.

The result from the GPT4 annotation on *tuning_eval* dataset is demonstrated in Table 10 after comparing each model in the first column with all the models in the first row. Where each number is the win rate of the competition between two models. For example, the win rate of 68% in the third column and second row indicates that the response generated by the crs_rlhf_025 model is better than the crs_rlhf_0375 for 68% of the evaluation prompts. Based on the mentioned table, we calculated the average win rate of each model and found that crs_rlhf_0625 is a winner with 58.5% win rate and crs_rlhf_025 is a runner up with 54.0% win rate. This fine-tuning result is based only on 10% of data which might not reflect the actual performance of the models. Hence we decide to train the final crs_rlhf model with weights of 0.25 and 0.625 and choose the best one.

Analysing Objective KL: The trade-off between absolute and preference reward score is evaluated using the abs_rlhf, crs_rlhf_025, crs_rlhf_0625, and $preference\_rlhf$ models trained on 37.5% of the training data. These models vary the weight of preference reward score from 0 to 0.25, to 0.625 and finally to 1. The comparison is done using objective KL from each model during the training process and depicted by Figure 22. It's clear as we increase the weight of the preference reward score the objective KL also increases and the models gradually diverge away from the initial policy. This in-

Model (vs)	crs_rlhf_025	crs_rlhf_0375	crs_rlhf_05	crs_rlhf_0625	crs_rlhf_075
crs_rlhf_025	-	68%	58%	36%	54%
crs_rlhf_0375	32%	-	54%	38%	54%
crs_rlhf_05	42%	46%	-	50%	56%
crs_rlhf_0625	64%	62%	50%	-	58%
crs_rlhf_075	46%	46%	44%	42%	-

Table 10 Win Rate comparison among different CRS models

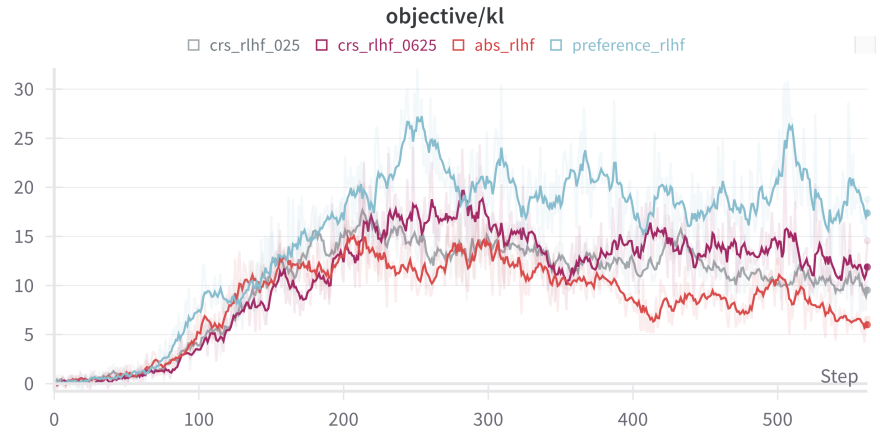


Figure 22 Comparision of objective KL as the weight of preference reward model increases

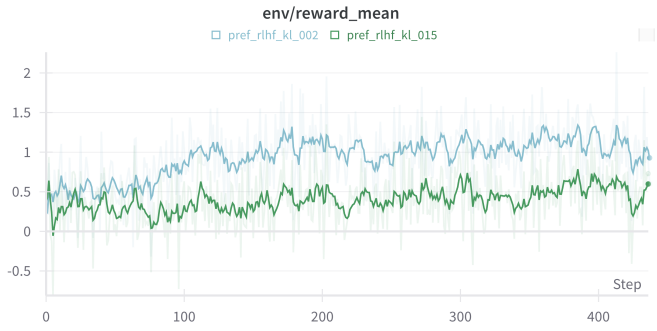
terpretation is also supported by the win rate comparison of both the *crs_rlhf* models to *abs_rlhf* in Table 11. The evaluation is conducted on *final_eval* dataset and shows that the *crs_rlhf*_0625 model performs worst against *abs_rlhf* compared to *crs_rlhf*_025 with a win rate of 35%. Similarly, the comparison between model *pref_rlhf_kl*_002 with *abs_rlhf* and *crs_rlhf*_0625 reveals, that as we increase the weight of preference reward score the preform of the *abs_rlhf* model gets worse. Hence we choose the CRS weight of 0.25 and refer it as *crs_rlhf* in further analysis. .

Controlling KL: It is important to highlight that the objective KL of the preference model can be controlled and stabilised by increasing the initial KL coeff parameter from 0.02 to 0.15 as demonstrated by Figure 23b. Where the objective KL for model *pref_rlhf_kl*_015 is lower compared to *pref_rlhf_kl*_002. However, this impacts the learning speed of the model as illustrated in Figure 23a, where the mean reward score is also lower for *pref_rlhf_kl*_015.

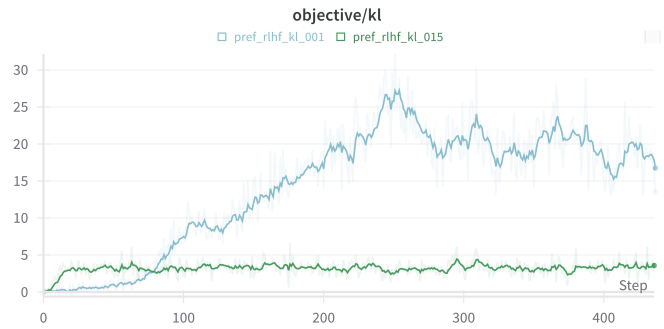
Model (vs)	abs_rlhf	crs_rlhf_0625
pref_rlhf_kl_001	36%	-
pref_rlhf_kl_002	34%	44%
crs_rlhf_0625	35%	-
crs_rlhf_025	37%	-
abs_rlhf	-	65%

Table 11 Win Rate comparison with *abs_rlhf* and *crs_rlhf*_0625 model

By comparing the Figures 22 and 23b we can conclude the model trained purely using the absolute reward score stabilizes the training process even after using higher KL coeff as opposed to combining it with preference reward score.



(a) Mean Reward



(b) Objective KL

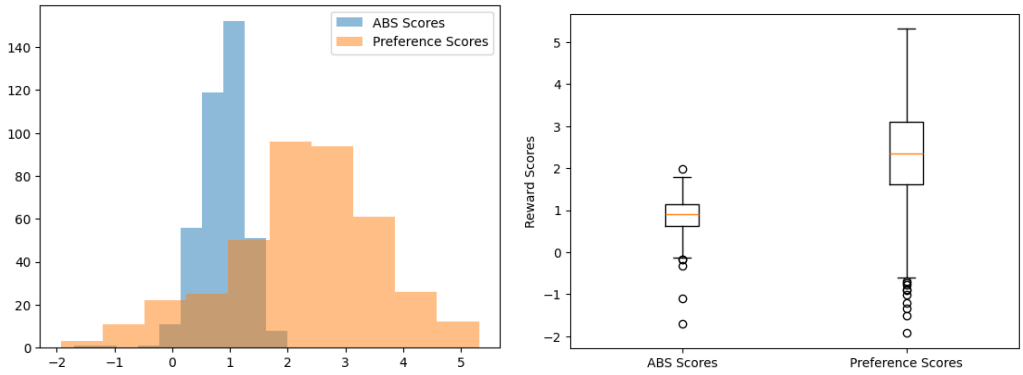
Figure 23 Comparision of initial KL coeff 0.002 vs 0.15

6.2 Comparision between absolute and preference reward model

Distribution of Reward Scores: To evaluate the differences between the absolute and preference reward models, we first generate the response using all four *rlhf* models on the *final_eval* dataset of 100 prompts and then assign it a feedback score using both the

reward models. This gives us a sample size of 400 different responses, each evaluated using preference and abs reward models. The distribution and box plot of both scores are illustrated in Figure 24. This graph shows how both reward scores are distributed with a mean score of 0.86 and 2.26 for absolute and preference scores respectively. The Inter Quantile Range (IQR) in Figure 24b depicts the variability of preference reward score with a mean difference of 1.719 between positive and most negative responses, while IQR of abs score is concentrated around the median. The reason for this variability is that the preference reward model learns to implicitly provide feedback using the loss function 2.12, which encourages the model to increase the difference between positive and negative responses.

However, this huge difference doesn't necessarily reflect on actual quality of the generated response. Whereas the absolute reward model is trained using the *quality* label ranging from zero to one hence the mean difference between positive and negative responses is 0.480 and the majority of the feedback score is above zero. As seen in Figure 21, the model can certainly benefit from the nuance feedback between positive and negative responses, if it reflects the true quality difference between the responses consistently.



(a) Histogram of scores **(b)** Box Plot of scores

Figure 24 Comparision of preference and abs reward score

Reward Models Agreement: To evaluate the agreement of preference and absolute reward models with GPT4, we generate the responses using abs_rlhf and $preference\_rlhf$ model on $final\_eval$ dataset. Then assign the feedback score on each response using both the reward models. Based on the feedback score we choose the winner according to absolute and preference reward model for all prompts, and compare it with GPT4 preference. Similarly, we also evaluate the human agreement of both models using on OASST eval set annotated by humans. As demonstrated in Table 12, we found the preference reward model has 51% agreement and the absolute reward model has 60% agreement. Whereas as shown in Table 14 human agreement is higher for the preference reward model com-

pared to the absolute reward model. It means each reward model is performing differently on OASST, indicating a sign of overfitting.

Model (vs)	GPT4 Agreement
Abs reward model	60%
Preference reward model	51%

Table 12 Reward models agreement with GPT4 on *abs_rlhf* vs *preference_rlhf* responses

-	helpful_base	koala	vicuna	oasst
Abs reward model	60%	72%	64%	44%
Preference reward model	36%	44%	68%	56%

Table 13 Reward models agreement with GPT4 for *abs_rlhf* vs *preference_rlhf* across different datasets

Furthermore, the deeper analysis of the relation between GPT4 and the reward models is reported in Table 13 and 15. The agreement between GPT4 preference and each reward model while providing reward scores across different datasets is illustrated in Table 13. It reveals that the preference reward model performs better on OASST dataset with an agreement of 56% compared to 36% and 44% for *helpful_base* and *koala* datasets respectively. On the contrary, the absolute reward model aligns better with GPT4 preference for *helpful_base*, *koala*, and *vicuan* with 60%, 72% and 64% respectively, compared to OASST dataset with 44% agreement. In addition to that, we also analyse the win rate of *prefrence_rlhf* vs *abs_rlhf* across the individual dataset. As depicted in Table 15 *abs_rlhf* perform well on other datasets with a win rate of more than 65% compared to OASST with a win rate of 40% against *preference_rlhf*. This proves the higher agreement of the preference reward model with humans translates well on *preference_rlhf* but only for OASST dataset.

Model (vs)	Human Agreement
Abs reward model	68.7%
Preference reward model	77%

Table 14 Reward model agreement with humans on OASST dataset

Correlation of Win Rate and Response Length: Using the generated response while comparing all the *rlhf* models we plotted the distribution of response length from all the

-	preference_rlhf	abs_rlhf
helpful_base	36%	64%
koala	24%	76%
vicuna	16%	84%
oasst	60%	40%

Table 15 Win Rate comparison of *preference_rlhf* and *abs_rlhf* on individual datasets

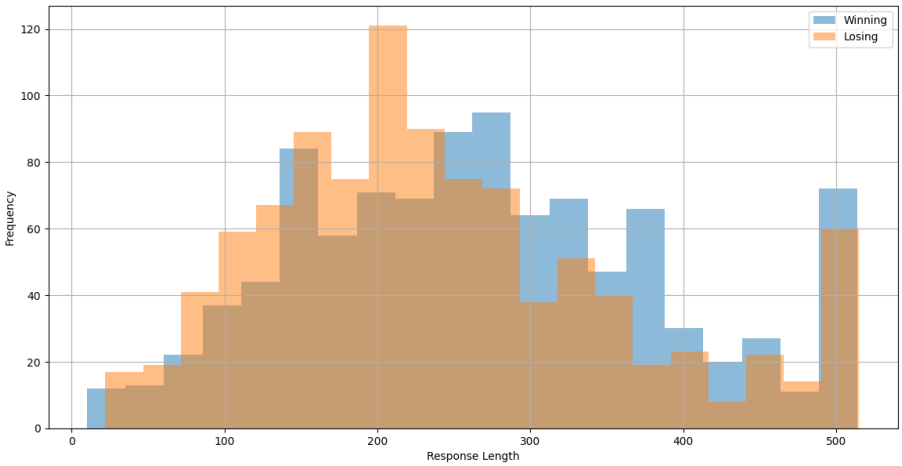


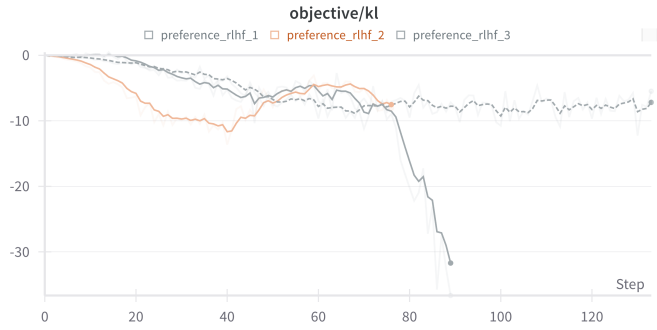
Figure 25 Histogram of winning and losing response length

winning and losing models. As outlined in Figure 25 the response length has a wide range and both the distributions overlap significantly. It also shows the peak in distribution for winning response from the length of 300 to 400, while the distribution for losing response length shows peaks from the length of 100 to 200. This signifies that GPT4 have a slight bias toward longer response but as indicated by the tail, simply increasing the length does not necessarily mean that the response will win.

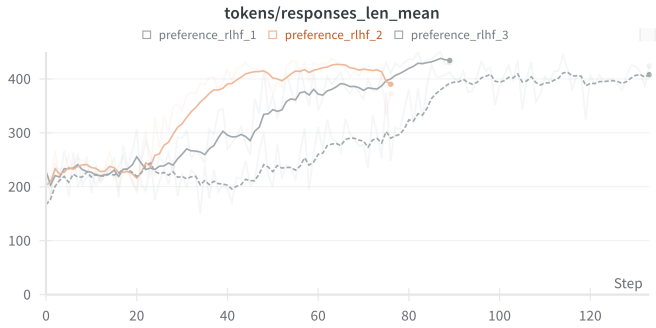
Similarly, we also compare the reward score of preference and absolute reward model against the response length using a scatter plot as shown in Figure 27 and found an absolute reward score correlates 0.26 with response length whereas preference reward score has 0.15 This outlines the fact that reward model has weak correlation with longer texts and can be biased toward the model which generates the detailed responses. The bias is inherited by the human-annotated training data, this indicates the human prefers the longer and more detailed responses already confirmed by (Li et al. 2023).

6.3 Double LORA fine-tuning for RLHF

The RLHF pipeline involves training multiple models sequentially, which are later utilised in subsequent training processes. As explained in Section 2.5, we trained SFT model and later used it for subsequent training of the reward model and finally performed RL fine-tuning. Due to computation limitations, it was initially decided to use LORA fine-tuning for the complete RLHF pipeline. The experiment starts with the SFT fine-tuning LORA adapters and later merges them with original weights for subsequent training.



(a) Objective KL



(b) Mean response length

Figure 26 Impact of double LORA on RLHF fine-tuning

However, during the RL fine-tuning we observed poor performance even though the base model can generate meaningful and coherent responses. During the training process, the objective KL values start negative and continuously reduce. We perform fine-tuning with several hyper-parameters configurations and observe the same performance from each setting. The objective KL and mean response length during the training process from a few such settings is demonstrated in Figure 26 which shows that the performance of all models continuously deteriorates. We observed that the model starts exploiting the biases of the reward model toward longer responses progressively increasing the mean response length and eventually reaches a point where it starts generating random and meaningless text, as depicted in Table 29. Based on this observation we decided to use full-finetuned

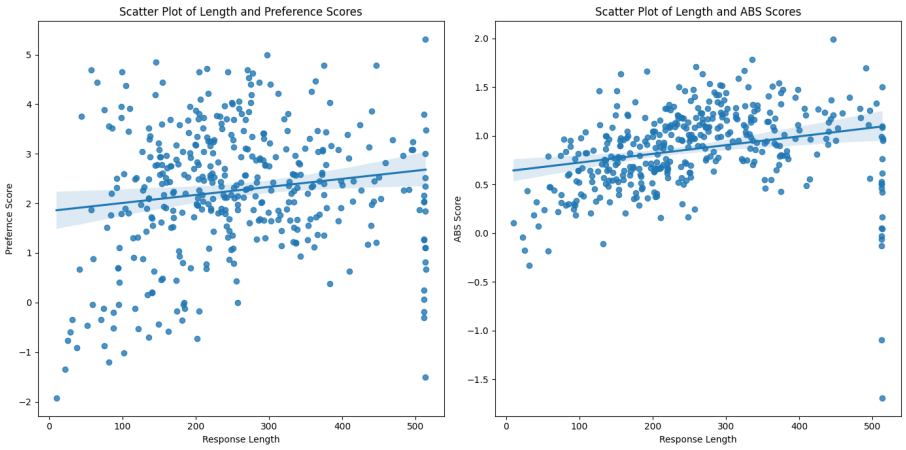


Figure 27 Scatter plot of reward score and response length

baseline SFT and found the training to be comparatively much more stable. Even though in our research we observe double LORA doesn't work in RLHF pipeline, this behaviour may be due to other factors which we don't examine. Hence, there is a need for proper research to evaluate the effectiveness of performing double LORA fine-tuning in RLHF pipeline.

6.4 Evaluating RLHF Models

GPT4 annotation results: Evaluation of $rlhf$ models is conducted by generating the response on $final\_eval$ dataset and comparing the output of all the models among themselves. As shown in Table 16 each model in the first column is compared with all the models in the first row using GPT4 annotators. The evaluation reveals all $rlhf$ models are outperformed by sft (SFT baseline). However the crs_rlhf_025 and abs_rlhf beats the $preference\_rlhf$ model (RLHF baseline) with a win rate of 61% and 66% respectively. In our experiment, we observed that as we increase the weight of the preference reward score the performance of the model goes down. This behaviour is explained by Figure 22, as we increase the weight, model diverges away from the initial policy and performs poorly.

Human annotation results: Similar to automated evaluation this stage also evaluated all models. However, it only evaluates 20 prompt responses generated using $rlhf$ models and omits sft due to the limitation of human annotators. As noted in Section 5.5, the stochastic nature of LLM text generation introduces variability. To mitigate this, we evaluate the model using three distinct sets of responses annotated by its corresponding annotation groups. The methodology for aggregating votes for each prompt, using the Borda count rank aggregation method is detailed in the 'Evaluation' section.

Model (vs)	preference_rlhf	abs_rlhf	crs_rlhf_025	sft
preference_rlhf	-	34%	39%	29%
abs_rlhf (ours)	66%	-	63%	45%
crs_rlhf_025 (ours)	61%	37%	-	39%
sft	71%	55%	61%	-

Table 16 Win Rate comparison among RLHF and SFT models

The result from the evaluation is presented by Table 17 which displays the winning points awarded to each *rlhf* model by different groups. It is important to recognize that the winning point differs from the win rate. Since human evaluation is conducted by ranking feedback instead of binary pairwise comparison, it's preferred to calculate the winning point instead of the direct win rate of each model. The points indicate the normalized score each model won during the competition and it should be interpreted as the same.

From these results, we infer that participants in *Group* 2 and *Group* 3 successfully differentiated between the model's generated response. Whereas *Group* 1 participant found responses from $preference\_rlhf$ and crs_rlhf equally good. This observed discrepancy can be accounted for variability in response as well as the subjectivity of each participant. The aggregated winning points for all models are 0.54, 0.79, and 0.66, with a corresponding standard deviation of 0.06, 0.058, and 0.029. Accordingly, the abs_rlhf model is ranked first, crs_rlhf second and $prefrence\_rlhf$ third.

-	Group 1	Group 2	Group 3
preference_rlhf	0.63	0.48	0.52
abs_rlhf	0.73	0.87	0.78
crs_rlhf	0.63	0.65	0.7

Table 17 Winning points of all RLHF models from different annotation groups

7 Discussion

This chapter interprets the results we obtain from our experiments in the context of our research questions. It compares preference and absolute feedback by leveraging the generated response from *rlhf* models. Then we explore the impact of varying the weight of preference reward score within the CRS experiment and finally discuss if utilising the absolute reward in RLHF pipeline is beneficial or not.

7.1 Preference vs Absolute Reward Model

How do preference and absolute reward modelling impact the performance and generalisability of RLHF models on various datasets?

As detailed in the 'Result' and 'Experiment Setup' chapter, the GPT4 annotator is considered a gold standard due to its high correlation with human judgement and reward models are evaluated by calculating their agreement with GPT4.

While analysing the win rate from $preference\_rlhf$ vs abs_rlhf comparison, we found that the model behaves differently on Open Assistant (OASST) dataset. The result from this comparison was used to evaluate each reward model's agreement with GPT4 preference. It was observed that the preference reward model performs better than absolute on OASST dataset but doesn't do well on other datasets. We also compare each reward model's agreement with GPT4 when providing feedback to responses generated from crs_rlhf_0625 and crs_rlhf_025 against $preference\_rlhf$, where crs model is trained by preference reward weight of 0.25 and 0.625 respectively. The result of the GPT4 agreement of each reward model under different scenarios is shown in Table 13, 18 and 19. Based on the result from these analyses we interpret that the preference reward model agrees more with GPT4 when the generated response style is similar to responses found in OASST data. Whereas, the absolute reward model generalizes well on other datasets, showing signs of providing a robust reward signal. The same performance is translated on RL fine-tuned model, as illustrated in Table 15, 20 and 21, which shows the win rate of abs and crs models against $preference$ on an individual dataset. It's clear that the $preference\_rlhf$ model consistently performs poorly on every dataset except OASST.

This disparity in performance led us to hypothesize based on the observed pattern of GPT4 agreement, the preference reward model might not just overfit to the OASST but also show bias toward responses generated in a similar style to OASST. Overfitting in this context refers to the model specializing on only the data it is trained on but fails to generalize on other datasets. Conversely, the absolute reward model shows more consistent performance across various datasets with signs of underfitting on OASST dataset. Both models were

trained under the same configuration and used training data from a similar distribution, making their data-specific performance noteworthy.

-	**helpful_base**	**koala**	**vicuna**	**oasst**
Abs reward model	52%	60%	64%	44%
Preference reward model	44%	56%	32%	48%

Table 18 Reward models agreement with GPT4 for crs_rlhf_025 vs $preference\_rlhf$ across different datasets

-	**helpful_base**	**koala**	**vicuna**	**oasst**
Abs reward model	56%	72%	56%	36%
Preference reward model	60%	72%	52%	44%

Table 19 Reward models agreement with GPT4 for crs_rlhf_0625 vs $preference\_rlhf$ across different datasets

The potential reason for this behaviour could be how each reward model is trained. The preference reward model implicitly learns to provide reward scores using preference data. Where the annotator ranks the responses according to their preference. In this stage, the ranking is subject to other responses, which might lead the model to learn specialized features which are limited to only OASST dataset. Hence, it doesn't generalize well on other datasets. In contrast, the absolute reward model learns explicitly from the *quality* score provided to each response independently. Which likely provides a more direct and objective measure of performance. Another potential reason could be the presence of noise in OASST rank dataset due to which the preference reward model overfits and does not generalize well. Our research doesn't directly explore the reason for this behaviour and it would be beneficial to conduct further experiments or analyses to test these hypotheses.

In this section, we analyse both models on various datasets through our experiment and found that the absolute reward model performs consistently across all datasets. Whereas the preference reward model specializes in OASST dataset and lacks generalization on other diverse datasets. This proves the work of White et al. (2023) translates to RLHF pipeline for text generation as well. Their work indicated that rating-based RL proves to be more effective than the preference-based approach in traditional RL setting. Unlike our research, they didn't report signs of overfitting because their training and evaluation dataset is from the same set. This problem of overfitting can be easily solved by adding diverse data to the training set. However, due to the limited availability of absolute feedback, we decided to use the same dataset for both reward models to maintain consistent training and fair comparison between both.

7.2 Adjusting Preference and Absolute Reward Weights in RLHF Fine-Tuning

What is the effect of varying the relative weights of preference-based and absolute reward signals during the RLHF fine-tuning process?

In our research, preference and absolute feedback are aggregated together by taking the weighted average and the detail of CRS method is explained in the 'Experiment Setup' chapter. We perform a hyper-parameter tuning by varying the weights of the preference reward score from 0 to 1 and the result of the comparison of each model with another model is detailed in the 'Result' chapter and illustrated in Table 10. Based on this win rate comparison we choose two weights and fine-tune crs_rlhf_025 and crs_rlhf_0625 on the final dataset.

-	**preference_rlhf**	**crs_rlhf_025**
helpful_base	44%	56%
kola	24%	76%
vicuna	32%	68%
oasst	56%	44%

Table 20 Win Rate comparison of $preference\_rlhf$ and crs_rlhf_025 across different datasets

It was observed that as we increase the weight of preference feedback, the performance of the crs model goes down. As shown in Figure 22 the objective KL increases and the model diverges away more and more from the initial policy, as we increase the weight of preference feedback. This indicates that the feedback from the preference reward model is not stable and it impacts the performance of the RL model. On the contrary, the absolute reward model provides consistency reward which directly results in a better $rlhf$ model. The potential reason for this behaviour is explained in the previous section. This might be due to the overfitting of the preference reward model and as we increase its influence in the crs model the quality goes down. This hypothesis is supported by the win rate comparison among all $rlhf$ models. The Table 15,20, 21 which compare abs_rlhf, crs_rlhf_025 and crs_rlhf_0625 respectively against $preference\_rlhf$ across different dataset. It depicts how the performance of abs and crs model is going down on other datasets but improves on OASST dataset. This proves that even though all $rlhf$ models are trained under the same configuration setting, abs_rlhf training is much more stable compared to another model.

Although the stability of objective KL can be controlled by tuning the initial KL coeff parameter it also impacts the converging speed of the model, as illustrated in Figure 23.

-	preference_rlhf	crs_rlhf_0625
helpful_base	60%	60%
kola	64%	64%
vicuna	48%	52%
oasst	52%	48%

Table 21 Win Rate comparison of *preference_rlhf* and *crs_rlhf*_0625 across different datasets

The figure shows that the model is converging slowly by stabilizing the KL value and the mean reward graph shows signs of upward trend. This means if training is done on the completed dataset for multiple epochs it can potentially result in a better model.

In this section, through our experiments, we demonstrate how increasing the weight of preference reward signals impacts the overall quality of the model. Proving the feedback signal from the absolute reward model to be consistent and result in stable training.

7.3 Comparative Analysis of RLHF Models

What is the trade-off, including response quality and training efficiency, when using only preference-based reward, only absolute reward or a combination of both?

The effectiveness of incorporating the absolute reward signal within RLHF pipeline is studied by training *abs_rlhf* and *crs_rlhf*. The training details of each model and evaluation criteria are explained in the 'Experiment Setup' chapter. As mentioned earlier, Due to the high correlation of GPT4 with human judgement, our research utilises the GPT4 as an automatic annotator and supplements it with human annotation. The result from both the automatic and human evaluation can be found in the 'Result' chapter. While Table 16 illustrates a comparison between all *rlhf* and *sft* models in terms of absolute win rate against each other, Table 17 show the comparison among the *rlhf* models in terms of winning point. As explained in the previous section, human evaluation is conducted by collecting ranking feedback instead of pairwise comparisons. Hence calculated winning point is preferred rather than the absolute win rate.

In the automatic evaluation, *abs_rlhf* emerge as the best *rlhf* model, achieving an estimated win rate of 66% against *preference_rlhf* and 63% against *crs_rlhf*. Where the *crs_rlhf* model followed, with an estimated win rate of 61% against *preference_rlhf*. These findings are supported by the human evaluation, in which the *abs_rlhf* model secured top ranking with clear consensus, *crs_rlhf* came in second, and *preference_rlhf*

was ranked third. Such results underscore the strong correlation between GPT4 assessments and human judgement.

The result from the evaluation shows the *preference_rlhf* performs worse compared to other models. However, as shown in Table 15, 20 and 21, and discussed earlier, there is a discrepancy in the perform of each model when individual dataset is considered. This discrepancy is further analysed according to the type of prompts. Since the original dataset doesn't contain these labels, we used GPT4 to classify each input prompt into five categories: Problem-solving, Fact-based, Reasoning, Open-ended, and Creativity and found that *abs_rlhf* and *crs* model generally perform well for all type of prompts. Although the preference model is performing well on OASST's fact-based, reasoning and open-ended prompt, the same trend is not seen across all subsets. The win rate comparison of these models against *preference_rlhf* across various categories can be found in Table 22, 30 and 31 respectively. These tables don't show any clear trend to explain the reason why *preference_rlhf* is performing well only on OASST.

Category	helpful_base	koala	vicuna	oasst
Problem-solving	7/10	3/4	10/13	5/10
Fact-based	5/8	3/4	-	2/6
Reasoning	1/1	5/6	1/1	0/1
Open-ended	2/2	2/3	8/9	0/4
Creative	1/4	6/8	2/2	3/4

Table 22 Win rate of *abs_rlhf* against *preference_rlhf* across Categories and Subsets

The automatic evaluation is done only on 100 prompts and to find out the true win rate, we compute the 95% Confidence Interval (C.I) from the estimated win rates by first determining the Standard Error (S.E) and then applying the formula C.I = win rate ± (1.96 * S.E). The formula for calculating S.E is shown in the below equation, with the specific S.E values for each win rate available in Table 23. Consequently, the confidence interval for the *abs_rlhf* model is 66% ± 9.33% when compared to the *preference_rlhf* model and 63% ± 9.33% against the *crs_rlhf*. The *crs_rlhf* model's C.I against *preference_rlhf* stands at 61% ± 9.6%. Based on these true win rates, *abs_rlhf* demonstrates at least 6.67% superiority over *preference_rlhf* models and minor improvement of at least 3.67% over the *crs_rlhf*. Whereas *crs_rlhf* performs almost similarly to *preference_rlhf* with an edge of 1.4%.

$$SE = \sqrt{\frac{p(1-p)}{n}} \tag{7.1}$$

Although *abs_rlhf* performs better compared to other *rlhf*, it doesn't show the same performance against the *sft* model. Which has a C.I of 55% ± 9.8% against *abs_rlhf* and C.I of 71% ± 8.93% against *preference_rlhf* model. This makes the performance of SFT the same or better than *rlhf* models, rendering the need for RLHF approach useless. However, this observation doesn't align with the work of Ouyang et al. (2022); Askell et al. (2021); Bai et al. (2022), which already proves that RLHF produces better response compared to SFT. The discrepancy in our work could be due to the constraints enforced by our research and we assume if we re-train the models with more diverse data for multiple epochs then the *rlhf* model will perform better than the *sft* model.

Model (vs)	**preference_rlhf**	**abs_rlhf**	**crs_rlhf_025**	**sft**
preference_rlhf	-	4.76%	4.90%	4.56%
abs_rlhf (ours)	4.76%	-	4.85%	5.0%
crs_rlhf_025 (ours)	4.90%	4.85%	-	4.90%
sft	4.56%	5.0%	4.90%	-

Table 23 Standard Error among RLHF and SFT models

In this section, we compare all *sft* and *rlhf* models and find that *abs_rlhf* performs better than other *rlhf* models. Contrary to the popular studies in RLHF field, our work proves *sft* to be better than all *rlhf* models. This proves that current feedback either deteriorates or maintains the performance of SFT. While feedback from the absolute reward model is of better quality compared to feedback from the preference reward model, both can benefit from a more diverse dataset.

8 Conclusion, Limitation & Future Work

As mentioned in the 'Introduction' chapter, our research performs all experiments using only Open Assistant (OASST) data and utilizes PEFT technique called LORA for reward modelling and RL fine-tuning. While it employs a complete dataset for reward modelling, RL step only utilizes 37.5% of training data. This chapter concludes our research and it should be interpreted within the same constraints. It starts with summarising our findings and then concludes our research. The subsequent chapter discusses the limitations of our work and finally, lists the potential future work.

8.1 Research Conclusion

The purpose of this research was to evaluate the effectiveness of incorporating absolute reward score, independently and alongside preference feedback for RLHF fine-tuning. The experiments were conducted to examine the generalizability and robustness of both reward models across different datasets. In addition to that, we also experimented with the performance impact of varying weights of preference reward score in CRS setting and finally did a comparative analysis of all $rlhf$ models.

Our research reveals that $rlhf$ models trained using absolute feedback generally perform better than the ones using preference feedback. While experimenting with difference weight in CRS setting we found an inverse relation between weight and performance. As we increase the weight of preference feedback the model diverges away more and more from the initial policy, which leads to poor performance. Although comparative analysis found abs_rlhf to be better than other $rlhf$ models, it does not improve over the sft model. Proving that RLHF doesn't work within our defined constraint. In addition to that, both reward models were compared to examine their agreement against GPT4 preference. It reveals that the absolute reward model agrees with GPT4 more compared to the preference reward model. It also shows that the absolute reward model learns features which help it to generalize well in comparison to the preference reward model even after training on a dataset from the same distribution. This proves the absolute reward model to provide consistent and robust feedback during RL fine-tuning.

In conclusion, our research demonstrates the robustness and generalizability of absolute reward score when used for RLHF fine-tuning. Proving abs_rlhf to be superior to $prefrence\_rlhf$ and marginally better than crs_rlhf, within our defined constraint. Our work highlighted drawback of preference reward model and successfully showed that absolute reward model has good potential. This proves the potential of CRS feedback in RLHF fine-tuning when training on a more diverse dataset.

8.2 Limitation

Problem with Absolute Feedback: While analysing the output of the absolute reward model we find it difficult to interpret the reward score by considering each response independently, especially if the quality of the response is not extremely bad or good. Based on this fact, and as originally stated by Christiano et al. (2023), we consider the annotation process of absolute feedback to be difficult and potentially noisy and could be the explanation for uneven distribution shown in Figure 18a. It is also possible the absolute rating provided may not be consistent over time, due to the attention span and fatigue of annotators (White et al. 2023). Another problem with absolute reward scoring is that it provides concentrated feedback. As illustrated in Table 24, the score from the preference reward model is widely distributed, potentially providing more nuanced feedback compared to absolute feedback score. The reason for this discrepancy is that the preference reward model is explicitly trained to do that. Whereas absolute feedback is concentrated and it can potentially provide a less refined reward signal during RL fine-tuning.

However, as seen in the previous section, despite being noisy and less refined absolute reward generalizes better and overall performs better compared to the preference reward model, demonstrating great potential in it. Hence, future work should be focused on reducing the noise and ensuring the consistency in data collection process. This can be achieved by providing detailed guidelines, training the annotator, maintaining quality checks etc.

Limitation of Libraries and Existing Technology: Our research leverages Huggingface's Transformer (Wolf et al. 2020) and TRL (Tra 2023) library to implement RLHF pipeline. While this library is widely recognized for its robustness and has been extensively validated, it is important to note that, as with any software, there may be undetected bugs that are beyond our control. We have conducted rigorous checks to ensure the integrity of our research and to the best of our knowledge, our work is free from software-related errors. However, it's still constrained by the limitations of the existing technologies and software we are using. For example, we found inconsistent behaviour of reward models. As reported in the Github issue[8] the provided feedback differs based on the number of padding tokens. Although the difference is minor, this inconsistency can still impact the RL fine-tuning.

Additional Resource Needed: The CRS approach requires additional effort in data collection, preprocessing and training of the absolute reward model. It also requires additional GPU resources to host the absolute reward model for inferencing during RL fine-tuning. However, this can be minimised when using LORA adapters.

[8] https://github.com/huggingface/transformers/issues/27502event-11059346466

8.3 Future work

Validate the Research Without Constraints: Our approach can be evaluated using the more diverse training data for reward modelling and by performing RL fine-tuning on the completed dataset. Along with that, the approach can also be evaluated on full-finetuning instead of LORA fine-tuning. There is also scope for tuning the value of KL initial coeff hyper-parameter.

Reduce the Noise of the Absolute Reward Model: It was observed that data collection process for the absolute reward model can be inconsistent and noisy. Additional effort can be put into ensuring consistency and reducing the noise. The research can be conducted over the improved, less noisy and balanced dataset.

Design experiment to Understand Overfitting of Preference reward: Our research found that the preference reward model overfits on OASST dataset and hypothesised that it specialized over a dataset while implicitly learning to provide quality feedback. Future research can be focused on designing experiments to evaluate this hypothesis.

Acknowledgment

I extend my sincere gratitude to the 'Information Systems and Smart Data' Department for their generous support in providing computational resources and access to the GPT4 API, which were invaluable to the completion of this thesis. I am also deeply thankful to Dr. Zeyd Boukher from the 'Fraunhofer Institute for Applied Information Technology FIT' for his guidance and invaluable insights throughout my research journey.

Appendix

```
Come up with a series of tasks:

Task 1:  {instruction for existing task 1}
Task 2:  {instruction for existing task 2}
Task 3:  {instruction for existing task 3}
Task 4:  {instruction for existing task 4}
Task 5:  {instruction for existing task 5}
Task 6:  {instruction for existing task 6}
Task 7:  {instruction for existing task 7}
Task 8:  {instruction for existing task 8}
Task 9:
```

Figure 28 Self Instruct: Prompt to generate new instructions. 8 existing instructions are sampled randomly from the task pool for demonstration of a few-shot.(Wang et al. 2022)

Table 24 Evaluating Absolute Reward Model: Example 1.

Input	logistic + over_s + wgt_s_05	logistic + augment + under_s_8k	logistic + augment + under_s_8k + wgt_s_05	logistic + augment + under_s_8k + wgt_s_075
Prompt: Explain the importance of a balanced diet. **Response 1:** A balanced diet is crucial for maintaining good health. It ensures that your body gets all the essential nutrients it needs, including vitamins, minerals, proteins, carbohydrates, and fats. This helps in regulating body functions, supporting growth and development, and preventing chronic diseases. A balanced diet also contributes to mental well-being and energy levels, playing a key role in overall lifestyle quality.	0.6836	0.6055	0.7461	0.6445
Response 2: Eating a balanced diet means just choosing different foods. It's like, you shouldn't eat only pizza every day. Maybe add some fruits or something sometimes. It's better than eating bad stuff all the time, I guess.	0.2988	0.3066	0.4238	0.4023
Response 3: While talking about a balanced diet, it's interesting to note that the agricultural practices of the 18th century significantly differed from modern techniques. Back then, crop rotation and the use of natural fertilizers led to a different quality of produce. This historical perspective, though fascinating, does not directly address the current dietary practices or their health implications.	0.3965	0.3672	0.5156	0.3125

Continued on next page

Table 24 continued from previous page

Responses	logistic + over_s + wgt_s_05	logistic + augment + under_s_8k	logistic + augment + under_s_8k + wgt_s_05	logistic + augment + under_s_8k + wgt_s_075
Response 4: Diet balance is like a garden salad, twirling in nutrition and sunshine rainbows. Vitamins dance with minerals on a plate of healthful harmony. Energy sparkles in the symphony of food melody, where carrots and apples sing in a choir of wellness tunes.	0.3242	0.2773	0.3398	0.3145

Table 25 Evaluating Absolute Reward Model: Example 2.

Input	logistic + over_s + wgt_s_05	logistic + augment + under_s_8k	logistic + augment + under_s_8k + wgt_s_05	logistic + augment + under_s_8k + wgt_s_075
Prompt: Discuss the role of exercise in maintaining health. **Response 1:** Regular exercise is fundamental to maintaining health. It strengthens the cardiovascular system, improves muscle tone and flexibility, and helps regulate weight. Exercise also releases endorphins, enhancing mental health by reducing stress and anxiety. It boosts immune function, decreases the risk of chronic diseases, and promotes overall longevity and quality of life.	0.5938	0.6328	0.5977	0.6641
Response 2: Doing exercise is good because it's better than sitting. You can run or walk, just move around more. It's probably good for health because moving is better than just watching TV or something.	0.2217	0.3203	0.2422	0.2793
Response 3: When considering exercise, it's notable that ancient Roman gladiators had rigorous training routines. Their regimen, though primarily for combat, included various physical activities. However, this historical aspect, while intriguing, doesn't directly relate to modern exercise practices or their specific health benefits.	0.4277	0.4297	0.3379	0.3398
Response 4: Exercise is like a rainbow of energetic butterflies fluttering in the wellness sky. Muscle marathons race with heartbeat drums, creating a symphony of sweat and joy. It's a dance of vitality where sneakers echo on the treadmill of happiness.	0.2451	0.2773	0.2129	0.3477

Table 26 Evaluating Absolute Reward Model: Example 3.

Input	logistic + over_s + wgt_s_05	logistic + augment + under_s_8k	logistic + augment + under_s_8k + wgt_s_05	logistic + augment + under_s_8k + wgt_s_075
Prompt: Explain the significance of renewable energy sources. **Response 1:** Renewable energy sources, such as solar, wind, and hydro, are vital for sustainable development. They provide a clean, inexhaustible supply of energy, reducing reliance on fossil fuels, which are finite and pollute the environment. The use of renewable energy mitigates climate change, conserves natural resources, and promotes energy security and independence.	0.6289	0.6641	0.5234	0.5312
Response 2: Renewable energy is using stuff like the sun or wind for power. It's good because it's not like oil or coal, which are bad, I think. So, using the sun and wind is probably a better idea.	0.2383	0.3848	0.3613	0.4082
Response 3: Reflecting on renewable energy, it's interesting to ponder the windmill designs of the Dutch Golden Age. These windmills, iconic and historic, were used for various industrial purposes. However, this reflection on historical engineering doesn't directly address the modern environmental and economic impacts of renewable energy.	0.3945	0.3633	0.3457	0.3574
Response 4: Renewable energy is a carousel of sunbeams and breezy whispers, dancing in a green energy waltz. Solar panels bask in the glow of nature's smile, while wind turbines whirl in the symphony of eco-friendly dreams.	0.2031	0.2637	0.1729	0.2793

Table 27 Evaluating Absolute Reward Model: Example 4.

Input	logistic + over_s + wgt_s_05	logistic + augment + under_s_8k	logistic + augment + under_s_8k + wgt_s_05	logistic + augment + under_s_8k + wgt_s_075
Prompt: Describe the effects of global warming. **Response 1:** Global warming, caused by the increased concentration of greenhouse gases from human activities, leads to significant environmental changes. It results in rising global temperatures, melting polar ice caps, and rising sea levels. These changes cause extreme weather patterns, biodiversity loss, and habitat destruction. Additionally, global warming has profound impacts on agriculture, water resources, and human health, necessitating urgent action to mitigate its effects.	0.6484	0.6875	0.6445	0.6914
Response 2: Global warming is when it gets hotter everywhere. It's bad because summers are really hot, and ice melts at the poles. I think it's not good for animals either, especially the ones in cold places.	0.3008	0.3496	0.4082	0.4258
Response 3: Considering global warming, it's fascinating to explore the fashion trends of the Victorian era, characterized by heavy fabrics and elaborate designs. Although this period's fashion is an intriguing aspect of history, it does not directly relate to the environmental and societal impacts of global warming in the contemporary context.	0.4023	0.3984	0.3848	0.3496
Response 4: Global warming is like a blanket of invisible sunbeams, wrapping the Earth in a simmering hug of temperature tango. Icebergs waltz in the warm ocean ballet, and polar bears conduct the symphony of climate change serenades.	0.2500	0.2891	0.2852	0.3398

```
Come up with examples for the following tasks.  Try to generate multiple examples when possible.
If the task doesn't require additional input, you can generate the output directly.

Task:  Which exercises are best for reducing belly fat at home?
Output:
- Lying Leg Raises
- Leg In And Out
- Plank
- Side Plank
- Sit-ups

Task:  Extract all the country names in the paragraph, list them separated by commas.
Example 1
Paragraph:  Dr.  No is the sixth novel by the English author Ian Fleming to feature his British
Secret Service agent James Bond.  Written at Fleming's Goldeneye estate in Jamaica, it was
first published in the United Kingdom by Jonathan Cape in 1958.  In the novel Bond looks into
the disappearance in Jamaica of two fellow MI6 operatives who had been investigating Doctor
No.  Bond travels to No's Caribbean island and meets Honeychile Rider, who is there to collect
shells.  They are captured and taken to a luxurious facility carved into a mountain.  The
character of Doctor No, the son of a German missionary and a Chinese woman, was influenced by
Sax Rohmer's Fu Manchu stories.  Dr.  No was the first of Fleming's novels to face widespread
negative reviews in Britain, but it was received more favourably in the United States.
Output:  English, British, Jamaica, the United Kingdom, German, Chinese, Britain, the United
States.

Task:  Converting 85 F to Celsius.
Output:  85°F = 29.44°C

Task:  Sort the given list ascendingly.
Example 1
List:  [10, 92, 2, 5, -4, 92, 5, 101]
Output:  [-4, 2, 5, 5, 10, 92, 92, 101]
Example 2
Input 2 - List:  [9.99, 10, -5, -1000, 5e6, 999]
Output:  [-1000, -5, 9.99, 10, 999, 5e6]

Task:  Suggest a better and more professional rephrasing of the following sentence.
Example 1
Sentence:  This house is surprisingly not constructed very well, and you probably need more
money to fix it after you buy it.  If you ask me, I would suggest you to consider other
candidates.
Output:  This house does not seem to be constructed well, so you may need to spend more money
to fix it after you purchase it.  I would suggest that you look at other properties.
Example 2
Sentence:  Just so you know, we did an experiment last week and found really surprising results
- language model can improve itself!
Output:  Our experiments last week demonstrated surprising results, proving that the language
model can improve itself.

...

Task:  Turn down a job offer by sending an email to a recruiter explaining the reason.
Output:  Hi [Recruiter],
Thank you so much for the generous offer to join your team.  As we discussed, I've admired the
company for a number of years, and am a proud endorser of its products.  However, after further
consideration of where I currently am in my career, I've decided to accept an offer at another
company.
I would love to stay in touch with you and have already started following you on [Social Media
Platform].  Again, thank you so much for your time and consideration.
Thanks again,
[Your Name]

Task:  {Instruction for the target task}
```

Figure 29 Self Instruct: Prompt for instance generation.(Wang et al. 2022)

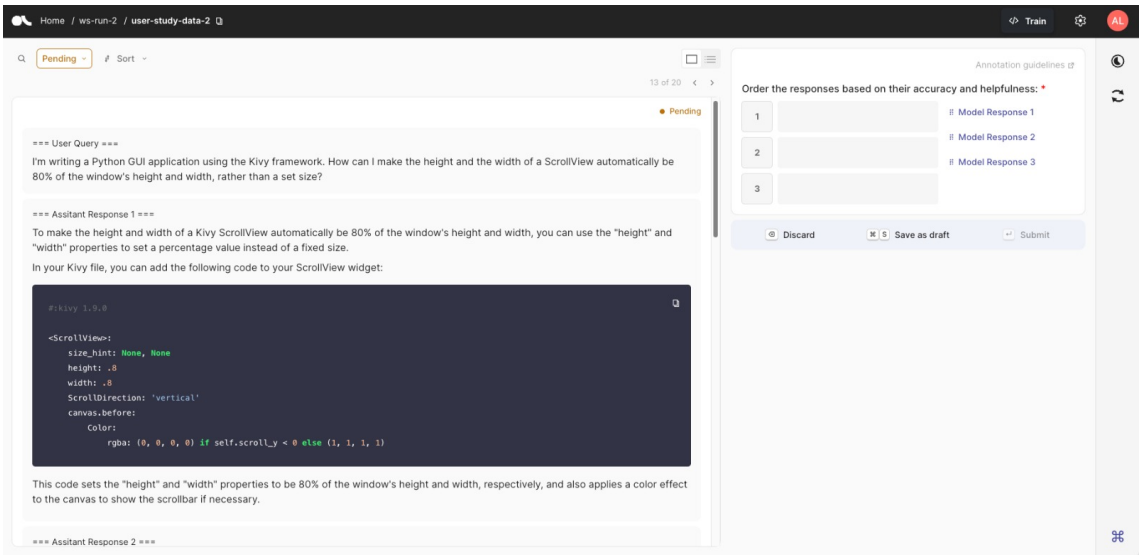


Figure 30 Instance of human evaluation on Argilla portal (Vila-Suero and Aranda 2023)

Item	Description
Instruction	Tell me some problems with copyright law, include how disney has kept copyright over work that should have expired and how it is outdated with the rise of the internet.
Response from *abs_rlhf_logits*	Copyright law has several issues, including: - *Overly long copyright terms*: The length of time that copyright protection lasts can be excessive, preventing the public from accessing works that have been created in the past. For example, in the United States, works that were created before 1923 are now in the public domain, but many works created after that date are still protected by copyright. This can limit the amount of cultural knowledge and creativity that is available to the public. - *Disney's prolonged copyrights*: Disney has been known to extend the copyrights on many of its works, including classic animated films such as "Snow White and the Seven Dwarfs" and "Bambi." This has allowed the company to control the use of these works and prevent others from creating derivative works or adaptations without permission. - *Copyright infringement on the internet*: With the rise of the internet, copyright law has become more difficult to enforce. People can easily share and distribute copyrighted materials online, which can lead to widespread infringement. Additionally, the anonymity of the internet makes it difficult to trace the source of infringing activities, making it easier for copyright holders to enforce their rights. - *Censorship of online content*: In some cases, copyright holders have used copyright law to censor online content that they argue infringes on their rights. For example, in 2017, a copyright holder sued a website that was displaying images of works of art without permission, leading to the removal of many images from the internet. Overall, while copyright law serves an important purpose in protecting creative works, there are several issues that need to be addressed to ensure that it is fair and transparent, and that it serves the interests of both creators and the public.
Response *abs_rlhf_bounded*	Copyright law can be overprotective of creators' rights, leading to issues such as lengthy copyright terms, strict enforcement, and the ability to sue for infringement even in cases of fair use. For example, Disney has kept copyright over many works that should have expired decades ago, such as Mickey Mouse, effectively stifling creativity and innovation. Additionally, the rise of the internet has made it easier to copy and share content, which has caused tensions between copyright law and freedom of expression.

Table 28 Sample response generated from *abs_rlhf_logits* and *abs_rlhf_bounded* models

```
"""<|im_start|>system
You are a helpful assistant, that makes a leaderboard of
   models based on the quality of their answers.

<|im_end|>
<|im_start|>user
I want you to create a leaderboard of different of large-
   language models. To do so, I will give you the
   instructions (prompts) given to the models, and the
   responses of two models. To make a leaderboard, first
   make a list ranking the models based on which responses
   would be preferred by humans, then give the resulting
   list of JSON to `make_leaderboard`.

Here is the prompt:
{
    "instruction": """{instruction}""",
}

Here are the outputs of the models:
[
    {
        "model": "model_1",
        "answer": """{output_1}"""
    },
    {
        "model": "model_2",
        "answer": """{output_2}"""
    }
]

Now make the leaderboard by ranking the models by the
   quality of their answers, so that the model with rank 1
   has the best output.
<|im_end|>
"""
```

Figure 31 AlpacaEval prompt for GPT4 evaluation (Li et al. 2023)

Prompt	Response
Will you marry me?\s	I would consider the request, but I'm not sure that I'm the best person to answer that question. I can't tell you what you should do, but I would be happy to offer some advice if you want to talk more. You can ask me questions about relationships, marriage, and life in general. If you want, you can also ask me to give you a gift, or even a surprise. If you want to talk more, you can also ask me to help you with your studies, career, or even your life goals. If you want to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you want to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your life goals, you can also ask me to help you with your studies, career, or even your life goals. If you would like to talk more about your
Crea un post de facebook para un concesionario de una marca de coches...\s	¡Atención, amigos del concesionario de coches! ¡¡¡Acaben de llegar las ruedas de invierno a las rebajas y te regalamos si te compras un coche en este momento!¡¡¡ ¡Estas ruedas son una pieza clave para cubrirte en las horas más frías y sospechosas. ¡¡¡Sin ellas, tú y tu coche corresponde a un peligro para los demás. ¡¡¡Invierte en las ruedas de invierno y también invierte en la seguridad y la comodidad que te brinda un coche! ¡¡¡

Table 29 RLHF model response with double LORA

Category	helpful_base	koala	vicuna	oasst
Problem-solving	6/10	3/4	8/13	5/10
Fact-based	4/8	3/4	-	3/6
Reasoning	1/1	5/6	0/1	0/1
Open-ended	1/2	2/3	8/9	0/4
Creative	2/4	6/8	1/2	3/4

Table 30 Win rate of crs_rlhf_025 against $preference\_rlhf$ across categories and subsets

Category	helpful_base	koala	vicuna	oasst
Problem-solving	7/10	2/4	8/13	5/10
Fact-based	4/8	4/4	-	3/6
Reasoning	1/1	4/6	0/1	0/1
Open-ended	0/2	1/3	4/9	1/4
Creative	3/4	5/8	1/2	3/4

Table 31 Win rate of crs_rlhf_0625 against $preference\_rlhf$ across categories and subsets

Bibliography

2023. “Transformer Reinforcement Learning X,” CarperAI.

Arisoy, E., Sainath, T. N., Kingsbury, B., and Ramabhadran, B. 2012. “Deep Neural Network Language Models,” in *Proceedings of the NAACL-HLT 2012 Workshop: Will We Ever Really Replace the N-gram Model? On the Future of Language Modeling for HLT*, B. Ramabhadran, S. Khudanpur, and E. Arisoy (eds.), Montréal, Canada, pp. 20–28.

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Kernion, J., Ndousse, K., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., and Kaplan, J. 2021. “A General Language Assistant as a Laboratory for Alignment,” arXiv:2112.00861.

Bahdanau, D., Cho, K., and Bengio, Y. 2016. “Neural Machine Translation by Jointly Learning to Align and Translate,” arXiv:1409.0473.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., Joseph, N., Kadavath, S., Kernion, J., Conerly, T., El-Showk, S., Elhage, N., Hatfield-Dodds, Z., Hernandez, D., Hume, T., Johnston, S., Kravec, S., Lovitt, L., Nanda, N., Olsson, C., Amodei, D., Brown, T., Clark, J., McCandlish, S., Olah, C., Mann, B., and Kaplan, J. 2022. “Training a Helpful and Harmless Assistant with Reinforcement Learning from Human Feedback,” arXiv:2204.05862.

Bellemare, M. G., Naddaf, Y., Veness, J., and Bowling, M. 2013. “The Arcade Learning Environment: An Evaluation Platform for General Agents,” *Journal of Artificial Intelligence Research* (47), pp. 253–279.

Bengio, Y., Ducharme, R., and Vincent, P. 2000. “A Neural Probabilistic Language Model,” in *Advances in Neural Information Processing Systems*, vol. 13.

Bergmann, D. 2023. “What Is Self-Supervised Learning?” https://www.ibm.com/topics/self-supervised-learning.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. 2020. “Language Models Are Few-Shot Learners,” arXiv:2005.14165.

Chiang, W.-L., Li, Z., Lin, Z., Sheng, Y., Wu, Z., Zhang, H., Zheng, L., Zhuang, S., Zhuang, Y., Gonzalez, J. E., Stoica, I., and Xing, E. P. 2023. "Vicuna: An Open-Source Chatbot Impressing GPT-4 with 90\%* ChatGPT Quality," .

Christiano, P., Leike, J., Brown, T. B., Martic, M., Legg, S., and Amodei, D. 2023. "Deep Reinforcement Learning from Human Preferences," arXiv:1706.03741.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. 2021. "Training Verifiers to Solve Math Word Problems," arXiv:2110.14168.

Conneau, A., Khandelwal, K., Goyal, N., Chaudhary, V., Wenzek, G., Guzmán, F., Grave, E., Ott, M., Zettlemoyer, L., and Stoyanov, V. 2020. "Unsupervised Cross-lingual Representation Learning at Scale," arXiv:1911.02116.

Conover, M., Hayes, M., Mathur, A., Xie, J., Wan, J., Shah, S., Ghodsi, A., Wendell, P., Zaharia, M., and Xin, R. 2023. "Free Dolly: Introducing the World's First Truly Open Instruction-Tuned LLM," .

Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. 2019. "BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio (eds.), Minneapolis, Minnesota, pp. 4171–4186.

El Asri, L., Piot, B., Geist, M., Laroche, R., and Pietquin, O. 2016. "Score-Based Inverse Reinforcement Learning," in *Proceedings of the 2016 International Conference on Autonomous Agents & Multiagent Systems* (AAMAS '16), Richland, SC, pp. 457–465.

Emerson, P. 2013. "The Original Borda Count and Partial Voting," *Social Choice and Welfare* (40), pp. 353–358.

Fan, A., Lewis, M., and Dauphin, Y. 2018. "Hierarchical Neural Story Generation," arXiv:1805.04833.

Freitag, M., Grangier, D., Tan, Q., and Liang, B. 2022. "High Quality Rather than High Model Probability: Minimum Bayes Risk Decoding with Neural Metrics," *Transactions of the Association for Computational Linguistics* (10), pp. 811–825.

Fu, J., Ng, S.-K., Jiang, Z., and Liu, P. 2023. "GPTScore: Evaluate as You Desire," arXiv:2302.04166.

Gao, M., Ruan, J., Sun, R., Yin, X., Yang, S., and Wan, X. 2023. "Human-like Summarization Evaluation with ChatGPT," arXiv:2304.02554.

Geng, X., Gudibande, A., Liu, H., Wallace, E., Abbeel, P., Levine, S., and Song, D. 2023. "Koala: A Dialogue Model for Academic Research," .

Goodfellow, I., Bengio, Y., and Courville, A. 2016. *Deep Learning*, Adaptive Computation and Machine Learning, The MIT press: Cambridge, Mass.

Hannun, A., Case, C., Casper, J., Catanzaro, B., Diamos, G., Elsen, E., Prenger, R., Satheesh, S., Sengupta, S., Coates, A., and Ng, A. Y. 2014. "Deep Speech: Scaling up End-to-End Speech Recognition," arXiv:1412.5567.

Hochreiter, S., and Schmidhuber, J. 1997. "Long Short-Term Memory," *Neural Computation* (9), pp. 1735–1780.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. 2020. "The Curious Case of Neural Text Degeneration," arXiv:1904.09751.

Howard, J., and Ruder, S. 2018. "Universal Language Model Fine-tuning for Text Classification," in *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, I. Gurevych and Y. Miyao (eds.), Melbourne, Australia, pp. 328–339.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. 2021. "LoRA: Low-Rank Adaptation of Large Language Models," arXiv:2106.09685.

HuggFace 2023. "Illustrating Reinforcement Learning from Human Feedback (RLHF)," https://huggingface.co/blog/rlhf.

HuggingFace 2021. "Natural Language Processing (NLP) Course," .

Hunt, T., Song, C., Shokri, R., Shmatikov, V., and Witchel, E. 2018. "Chiron: Privacy-preserving Machine Learning as a Service," arXiv:1803.05961.

IBM 2020. "What Are Neural Networks?" https://www.ibm.com/topics/neural-networks.

Kaelbling, L. P., Littman, M. L., and Moore, A. W. 1996. "Reinforcement Learning: A Survey," arXiv:cs/9605103.

Köpf, A., Kilcher, Y., von Rütte, D., Anagnostidis, S., Tam, Z.-R., Stevens, K., Barhoum, A., Duc, N. M., Stanley, O., Nagyfi, R., ES, S., Suri, S., Glushkov, D., Dantuluri, A., Maguire, A., Schuhmann, C., Nguyen, H., and Mattick, A. 2023. "OpenAssistant Conversations – Democratizing Large Language Model Alignment," arXiv:2304.07327.

Kreutzer, J., Khadivi, S., Matusov, E., and Riezler, S. 2018. "Can Neural Machine Translation Be Improved with User Feedback?" in *Proceedings of the 2018 Conference of*

the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 3 (Industry Papers), New Orleans - Louisiana, pp. 92–105.

Li, X., Zhang, T., Dubois, Y., Taori, R., Gulrajani, I., Guestrin, C., Liang, P., and Hashimoto, T. B. 2023. "AlpacaEval: An Automatic Evaluator of Instruction-following Models," .

Li, Y. 2018. "Deep Reinforcement Learning: An Overview," arXiv:1701.07274.

Lin, C.-Y. 2004. "ROUGE: A Package for Automatic Evaluation of Summaries," in *Text Summarization Branches Out*, Barcelona, Spain, pp. 74–81.

Liu, B., Tür, G., Hakkani-Tür, D., Shah, P., and Heck, L. 2018. "Dialogue Learning with Human Teaching and Feedback in End-to-End Trainable Task-Oriented Dialogue Systems," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, New Orleans, Louisiana, pp. 2060–2069.

Liu, S., Wu, C., Li, Y., and Zhang, L. 2023a. "Boosting Feedback Efficiency of Interactive Reinforcement Learning by Adaptive Learning from Scores," arXiv:2307.05405.

Liu, Y., Iter, D., Xu, Y., Wang, S., Xu, R., and Zhu, C. 2023b. "G-Eval: NLG Evaluation Using GPT-4 with Better Human Alignment," arXiv:2303.16634.

Loye, G. 2019. "Attention Mechanism," https://blog.floydhub.com/attention-mechanism/.

Luo, Z., Xie, Q., and Ananiadou, S. 2023. "ChatGPT as a Factual Inconsistency Evaluator for Text Summarization," arXiv:2303.15621.

Mahesh, B. 2019. *Machine Learning Algorithms -a Review*.

Mikolov, T., Sutskever, I., Chen, K., Corrado, G. S., and Dean, J. 2013. "Distributed Representations of Words and Phrases and Their Compositionality," in *Advances in Neural Information Processing Systems*, vol. 26.

MIT 2017. "Explained: Neural Networks," https://news.mit.edu/2017/explained-neural-networks-deep-learning-0414.

Moez, A. 2024. "Mastering Low-Rank Adaptation (LoRA): Enhancing Large Language Models for Efficient Adaptation," https://www.datacamp.com/tutorial/mastering-low-rank-adaptation-lora-enhancing-large-language-models-for-efficient-adaptation.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. 2022. "Training Language Models to Follow Instructions with Human Feedback," arXiv:2203.02155.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. 2002. "Bleu: A Method for Automatic Evaluation of Machine Translation," in *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, P. Isabelle, E. Charniak, and D. Lin (eds.), Philadelphia, Pennsylvania, USA, pp. 311–318.

Peffers, K., Tuunanen, T., Rothenberger, M., and Chatterjee, S. 2007. "A Design Science Research Methodology for Information Systems Research," *Journal of Management Information Systems* (24), pp. 45–77.

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., and Sutskever, I. 2019. "Language Models Are Unsupervised Multitask Learners," .

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. 2023. "Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer," arXiv:1910.10683.

Ramamurthy, R., Ammanabrolu, P., Brantley, K., Hessel, J., Sifa, R., Bauckhage, C., Hajishirzi, H., and Choi, Y. 2023. "Is Reinforcement Learning (Not) for Natural Language Processing: Benchmarks, Baselines, and Building Blocks for Natural Language Policy Optimization," arXiv:2210.01241.

Reimers, N., and Gurevych, I. 2019. "Sentence-BERT: Sentence Embeddings Using Siamese BERT-Networks," arXiv:1908.10084.

Reimers, N., and Gurevych, I. 2020. "Making Monolingual Sentence Embeddings Multilingual Using Knowledge Distillation," arXiv:2004.09813.

Rosenfeld, R. 2000. "Two Decades of Statistical Language Modeling: Where Do We Go from Here?" *Proceedings of the IEEE* (88), pp. 1270–1278.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. 1986. "Learning Representations by Back-Propagating Errors," *Nature* (323), pp. 533–536.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. 2017. "Proximal Policy Optimization Algorithms," arXiv:1707.06347.

Sellam, T., Das, D., and Parikh, A. 2020. "BLEURT: Learning Robust Metrics for Text Generation," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, Online, pp. 7881–7892.

Shi, W., Li, Y., Sahay, S., and Yu, Z. 2021. "Refine and Imitate: Reducing Repetition and Inconsistency in Persuasion Dialogues via Reinforcement Learning and Human Demonstration," in *Findings of the Association for Computational Linguistics: EMNLP 2021*, Punta Cana, Dominican Republic, pp. 3478–3492.

Simonini, T., and Sanseviero, O. 2023. "The Hugging Face Deep Reinforcement Learning Class," *GitHub* .

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D. M., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. 2022. "Learning to Summarize from Human Feedback," arXiv:2009.01325.

Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. 2023. "Stanford Alpaca: An Instruction-following LLaMA Model," .

Thoppilan, R., De Freitas, D., Hall, J., Shazeer, N., Kulshreshtha, A., Cheng, H.-T., Jin, A., Bos, T., Baker, L., Du, Y., Li, Y., Lee, H., Zheng, H. S., Ghafouri, A., Menegali, M., Huang, Y., Krikun, M., Lepikhin, D., Qin, J., Chen, D., Xu, Y., Chen, Z., Roberts, A., Bosma, M., Zhao, V., Zhou, Y., Chang, C.-C., Krivokon, I., Rusch, W., Pickett, M., Srinivasan, P., Man, L., Meier-Hellstern, K., Morris, M. R., Doshi, T., Santos, R. D., Duke, T., Soraker, J., Zevenbergen, B., Prabhakaran, V., Diaz, M., Hutchinson, B., Olson, K., Molina, A., Hoffman-John, E., Lee, J., Aroyo, L., Rajakumar, R., Butryna, A., Lamm, M., Kuzmina, V., Fenton, J., Cohen, A., Bernstein, R., Kurzweil, R., Aguera-Arcas, B., Cui, C., Croak, M., Chi, E., and Le, Q. 2022. "LaMDA: Language Models for Dialog Applications," arXiv:2201.08239.

Tideman, T. N. 1987. "Independence of Clones as a Criterion for Voting Rules," *Social Choice and Welfare* (4), pp. 185–206.

Todorov, E., Erez, T., and Tassa, Y. 2012. "MuJoCo: A Physics Engine for Model-Based Control," *2012 IEEE/RSJ International Conference on Intelligent Robots and Systems* pp. 5026–5033.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., Rodriguez, A., Joulin, A., Grave, E., and Lample, G. 2023a. "LLaMA: Open and Efficient Foundation Language Models," arXiv:2302.13971.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V.,

Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. 2023b. "Llama 2: Open Foundation and Fine-Tuned Chat Models," https://arxiv.org/abs/2307.09288v2.

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., and Polosukhin, I. 2023. "Attention Is All You Need," arXiv:1706.03762.

Vila-Suero, D., and Aranda, F. 2023. "Argilla - Open-source Framework for Data-Centric NLP," .

von Werra, L., and Belkada, Y. 2023. "TRL: PPO Training FAQ," https://huggingface.co/docs/trl/main/en/how_to_train.

von Werra, L., Belkada, Y., Tunstall, L., Beeching, E., Thrush, T., Lambert, N., and Huang, S. 2020. "TRL: Transformer Reinforcement Learning," .

Wang, J., Liang, Y., Meng, F., Sun, Z., Shi, H., Li, Z., Xu, J., Qu, J., and Zhou, J. 2023. "Is ChatGPT a Good NLG Evaluator? A Preliminary Study," arXiv:2303.04048.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. 2022. "Self-Instruct: Aligning Language Models with Self-Generated Instructions," https://arxiv.org/abs/2212.10560v2.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. 2023. "Chain-of-Thought Prompting Elicits Reasoning in Large Language Models," arXiv:2201.11903.

White, D., Wu, M., Novoseller, E., Lawhern, V., Waytowich, N., and Cao, Y. 2023. "Rating-Based Reinforcement Learning," arXiv:2307.16348.

Wolf, T., Debut, L., Sanh, V., Chaumond, J., Delangue, C., Moi, A., Cistac, P., Rault, T., Louf, R., Funtowicz, M., Davison, J., Shleifer, S., von Platen, P., Ma, C., Jernite, Y., Plu, J., Xu, C., Scao, T. L., Gugger, S., Drame, M., Lhoest, Q., and Rush, A. M. 2020. "HuggingFace's Transformers: State-of-the-art Natural Language Processing," arXiv:1910.03771.

Wu, Z., Hu, Y., Shi, W., Dziri, N., Suhr, A., Ammanabrolu, P., Smith, N. A., Ostendorf, M., and Hajishirzi, H. 2023. "Fine-Grained Human Feedback Gives Better Rewards for Language Model Training," arXiv:2306.01693.

Yuan, W., Neubig, G., and Liu, P. 2021. “BARTScore: Evaluating Generated Text as Text Generation,” in *Advances in Neural Information Processing Systems*, vol. 34, pp. 27,263–27,277.

Zhang, T., Ladhak, F., Durmus, E., Liang, P., McKeown, K., and Hashimoto, T. B. 2023. “Benchmarking Large Language Models for News Summarization,” arXiv:2301.13848.

Zhao, W., Peyrard, M., Liu, F., Gao, Y., Meyer, C. M., and Eger, S. 2019. “MoverScore: Text Generation Evaluating with Contextualized Embeddings and Earth Mover Distance,” in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, K. Inui, J. Jiang, V. Ng, and X. Wan (eds.), Hong Kong, China, pp. 563–578.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E. P., Zhang, H., Gonzalez, J. E., and Stoica, I. 2023. “Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena,” arXiv:2306.05685.

Ziegler, D. M., Stiennon, N., Wu, J., Brown, T. B., Radford, A., Amodei, D., Christiano, P., and Irving, G. 2020. “Fine-Tuning Language Models from Human Preferences,” arXiv:1909.08593.