Internship Project Report: Jiashu Xu

Intern: Jiashu Xu (jiashuxu)
Date: 8/23/2023

Title: Expanding Beyond Reward Models in Training Decoder Language Models
Abstract: Current reward models, which assess sentence textual quality, rely on shallow encoders that often struggle with generalization in multi-turn conversations. However, recent advancements with decoder-based LLMs, such as GPT-4, have demonstrated promising results in assessing text quality through simple prompting. Unfortunately, concerns around privacy and high query costs make this approach less than ideal for real-world applications. In this work, we show it's possible to finetune open-sourced decoder-only LLM on publicly-available datasets to match propriety LLMs and other existing SOTA. Furthermore, as benefits of decoder architecture, we show such reward model is able to zero-shot generalized to unseen dimensions and domain, boost performance via in context learning, facilitate effective data filtering, provide rationale to explain decisions and generate synthetic conversations for AI self-improvement.

Bio: Jiashu Xu is a master student at Harvard studying computer science. His research interest lies in efficient learning, ranging from learning from explanation to low-resource learning. Recently he is interested in LLM security and have worked on exploring the vulnerability of LLMs to instruction attacks. You can find more info at https://cnut1648.github.io/ or contact Jiashu Xu

Manager: Michael Johnston (mjohnstn)
Mentor: Daniel Pressel (djrp), Prasoon Goyal (prasog)
Onboarding Buddy: Luke Dai (dairluke)

Runbook: [OpenRewardModel] Runbook
Code branch: https://code.amazon.com/packages/AlexaTeacherModelExperiments/trees/heads/jiashuxu-open-rm/--/experiments/jiashuxu/open-rm

Note:
The report is organized in a paper-like way as I intend to write a paper on this project. For deliverables and project artifacts please refer to 3. Deliverables. To review the project and its story, feel free to skip 3. Deliverables.

1. Milestones

These milestones (originated and expanded from Plan: Open Reward Model) corresponds to the things achieved throughout the project and open future directions. Due to time constraint, I unfortunately leave some aspects little explored, but I encourage readers that are interested in this direction to try out.
These milestones correspond to things achieved throughout the project and open future directions.

- ☑ ~~Literature review on reward modeling~~
  - ☑ ~~OpenAI~~
  - ☑ ~~Anthropic~~
- ☑ ~~Cluster access and runbook~~
- ☑ ~~Data collection for reward modeling~~
  - ☑ ~~Following Shikib's work for comparsion~~
  - ☑ ~~Prompt format for data~~
- ☑ ~~Test run for 7B model~~
  ~~Set up training/evaluation pipeline~~
  - ☑ ~~Falcon~~
  - ☑ ~~GPT-J~~
  - ☑ ~~OpenLLaMA~~
- ☑ ~~Preliminary results~~
  - ☑ ~~Convert internal RAM datasets into preference datasets~~
  - ☑ ~~Zero-shot evaluation of OpenLLaMA, LLaMA, Falcon~~
    - ☑ ~~Use 7B, 13B, 30B and 70B~~
- ☑ ~~Set up baseline evaluation~~
- ☑ ~~Look for unseen metrics~~
  - ☑ ~~use internal datasets~~
- ☑ ~~Train larger scale model~~
  - ☑ ~~13B~~
  - ☑ ~~30B w/ LoRA~~
  - ☑ ~~30B w/o LoRA~~

  open source code and model

- ☑ ~~huggingface compatible training and inference~~
- ☑ ~~Match SOTA performance~~
  - ☑ ~~ICL inference~~

  Beyond RM
  - ☑ ~~generalize to new metric~~
  - ☑ ~~data filtering~~
    - ☑ ~~bedrock~~
    - ☑ ~~multi-turn bedrock~~

    rationale
    - ☑ ~~Generate rationale~~
    - ☐ Human Eval
    - ☐ Critic training

    Complete the cycle
    - ☑ ~~Create synthetic conversations~~
    - ☑ ~~Self-refine for 1 iteration~~
    - ☑ ~~Self-refine for 2 iterations~~
    - ☐ Self-refine for >2 iterations
- ☐ Final presentation
- ☐ Code documentation
- ☐ Project report

2. Introduction

Fig 1. A side-by-side comparison between classical encoder-based reward model that outputs a numerical score as proxy for conversation quality, and decoder reward model (this project) that outputs natural language which enables helpful properties such as **rationale generation** (5.2 Rationale Generation), **in-context learning with demonstrations** (Decoder enables In-Context Learning) and **preference dimension integration** (Evaluation prompt design with Preference Dimension). Colors correspond to that in the image on the left.
The goal of reward modeling is to model a particular (fine-grained) preference dimension. A reward model (RM) ingests a *conversation* (human, assistant, API, etc.) and assign a numeric *score* to a particular aspect/preference dimension (helpful, quality, instruction following etc.). Such score is useful in a wide array of applications, ranging from RLHF, rejection sampling to data filtering and text quality evaluation.

Current reward models, often encoder based, rely on shallow encoders that often struggle with generalization in multi-turn conversations. However, recent advancements with decoder-based LLMs, such as GPT-4, have demonstrated promising results in assessing text quality through simple prompting. Unfortunately, concerns around privacy and high query costs make this approach less than ideal for real-world applications.

In this work, we show it's possible to finetune open-sourced decoder-only LLM on publicly-available datasets to match propriety LLMs and other existing SOTA on a wide array of 9 diverse datasets across 16 unique preference dimensions. Furthermore, as benefits of decoder architecture, we show such reward model is able to zero-shot generalized to unseen preferences and domain, boost performance via in context learning, facilitate effective data filtering, provide rationale to explain decisions and generate synthetic conversations for AI self-improvement.

3. Deliverables

We release two models: LLaMA2 based 13B and LLaMA based 30B (download link in Decoder RM: [OpenRewardModel] Runbook) that is trained according to 4. Reward Modeling. Through extensive experiments on 6 commonly-used public benchmarks, 3 internal datasets and 16 unique preference dimensions, we show our decoder reward model consistently outperforms existing SOTA (Evaluation of absolute quality, Evaluation of relative quality) and matches the performance of proprietary LLMs such as ChatGPT, GPT4 or Claude 2 (Decoder Reward Model is on par with propriety LLMs). Decoder reward model is also helpful for internal use as well, e.g. filtering bedrock data (5. 1 Reward Modeling Filtering) that might go into SFT training data mixture.

We provide runbook ([OpenRewardModel] Runbook) in which detailed demonstrates: the model usage and sample script, code documentations, and instructions to reproduce model training. My mentor Dan and Prasoon already used the model internally, and people from other teams, such as Sina and Suhaila, also used the runbook to validate our bedrock filtering results (Alexa Prize Decoder Reward Model-based Human Demonstration Selection for AlexaLLM). Therefore I expect there should not be issues accessing and running code for wider audience. As we also plan to open-release the codebase, I have already inspected the code to ensure that there is no sensitive information or internal snippets.

4. Reward Modeling

Existing reward models either employs encoder-based models (Ethayarajh et al 2022, Köpf et al 2023) or replace the language model head with a linear head (Touvron et al 2023, Ganguli et al 2022, Askell et al 2021, Amazon Reward Modeling). Recently, studies on prompting LLM for text evaluation (LLM-Eval , G-Eval, GPTScore, inter alia) suggest that decoder models, potentially due to their extensive parameter size and the knowledge embedded via large-scale pretraining, may function as superior evaluators. Motivated by these findings, our study poses the

following research question: **Can a decoder-only model be trained directly for reward modeling?**

We have identified several advantages of using the decoder reward model:

- Eliminates the necessity to train a linear head, which has been empirically demonstrated to be challenging due to overfitting (Casper et al 2023, Xu et al 2023)
- Integrates evaluation and training within a single framework, specifically the decoder architecture and textual format.
- Offers a higher "soft upper bound," according to LLM evaluation works
- Conversations are often multi-turn, and encoder-based models are found difficult to adapt to such multi-turn conversations (Xu et al 2023, Ding et al 2023). On the contrary, decoder-only models maintain capacity to reason over long text (recent works show that it is possible to reason even up to 1,000,000,000 context size (Ding et al 2023)).

Formally, considering the conversation (possibly multi-turn)x and a preference dimension d that we care about (e.g. helpful, instruction-following), we first verbalize the two into natural language prompt v(d;x). Including dimension into the prompt leverages understanding and reasoning capacity of LLM to ground the output with respect to the target dimension d. Then we feed the prompt v(d;x) to a language model LM that outputs token sequence s1...n autoregressively. For the purpose of reward modeling, the prompt is written as QA style (e.g. "is the response helpful?"), so that we can use simple heuristic to judge the quality of x, e.g. by checking if LM outputs "Yes" or "No." Please refer to Evaluation prompt design with Preference Dimension for the exact prompts.
For the use case where a numeric score is preferred, e.g. data filtering (5. 1 Reward Modeling Filtering), we purpose to use logprobs as proxy. Concretely, for a generated token si, we compute logprob pi=lnPr(si|v(d;x),s<i)∈R<0 that represents model's confidence of next token being si conditioned on the prompt and all previous generated tokens. For outputs that model gives positive response (e.g. "Yes") we assign score of x as 1+minipi wheras −1+minipi for negative one (eg "No"). For on input, we use min logprobs among all generated tokens for the overall score of the input. Minimum logprob indicates model's least confidence parts which offers more signals.
Admittedly, such artificial scores could not form a continuous distribution (e.g. Fig 3 of ImageReward scores). This is due to the hard thresholds set at -1 and +1. However, such a scoring system is valuable since it facilitates a more straightforward assessment of absolute quality, eliminating the need for heuristic threshold selection or cases where determining a threshold is unfeasible (as seen with Amazon GPT-J).

4.1 Dataset Formulation

Data mixture

The lack of relevant human preference data (i.e., prompt distribution grounded on our taxonomy, rankings according to our guidelines, fine-grained preference dimension) has required us to largely rely on public datasets to train the reward model. To make fair comparison with Amazon internal reward model that is now widely-used by SFT and LHF team (Reward Modeling), we follow its training data (Alexa LLM LHF Dataset Tracker) with minimal modification.
Specifically, the data statistics for the training dataset and test dataset are reported in the following table:

Train Dataset	Subset	Dimension	Type	#instances		Test Dataset	Subset	Dimension	Type	#instar
Anthropic	base	helpful	Pairwise	39525		Anthropic	base	helpful	Pairwise	849
	online	helpful	Pairwise	20968			online	helpful	Pairwise	470
	rejection-sampled	helpful	Pairwise	48312			rejection-sampled	helpful	Pairwise	1056
WebGPT		helpful	Pairwise	11100		WebGPT		helpful	Pairwise	2747
Ethics	Commonsense	Commonsense	Pairwise	13910		Ethics	Commonsense	Commonsense	Pairwise	3885
	Deontology	Deontology	Pairwise	18164			Deontology	Deontology	Pairwise	3596
	Justice	Justice	Pairwise	21791			Justice	Justice	Pairwise	2704
	Utilitarianism	Utilitarianism	Pairwise	27474			Utilitarianism	Utilitarianism	Pairwise	9614
	Virtue	Virtue	Pairwise	28245			Virtue	Virtue	Pairwise	4975
SMF		summarization relevance	Pairwise	92382		SMF		summarization relevance	Pairwise	4304
OpenAssistant	rank	quality	Pairwise	20908		OpenAssistant	rank	quality	Pairwise	1025
	quality	quality	Pairwise	13366			quality	quality	Pairwise	677
	humor	humor	Pairwise	10693			humor	humor	Pairwise	549
	helpful	helpful	Pairwise	13925			helpful	helpful	Pairwise	691
	creative	creative	Pairwise	13879			creative	creative	Pairwise	666
10% SHP		interesting	Pairwise	325933		SHP		interesting	Pairwise	18409

For fair comparison between baselines (Baselines and Implementation Details), all datasets used in our project is of Pairwise type: each instance consists of triplet (`context`, `chosen`, `rejected`), where `context` is a single or multi-turn conversation between a human and an AI chatbot, `chosen` is one response generated by AI that human prefers over another generated response `rejected`.
In the following sections, we discuss how we formulate training and test instances into prompts such that can unlock full potential and elicit knowledges from LLMs. We note that this might not be the best prompt design, but thorough empirical studies (Robust to prompt design) we find that decoder reward model exhibits a degree of resilience to variations in the evaluation prompt. We leave finding the optimal evaluation prompt as future works.

Evaluation prompt design with Preference Dimension

As our model is decoder-only language model, it becomes intuitive to restructure the input into prompts. This allows us to harness the robust language comprehension and reasoning capabilities of the LLM. Moreover, we aim to build a reward model that supports multiple preference dimensions beyond helpfulness, and a natural approach to integrate dimension into model is to reformulate the prompt such that the final

prompt v is conditioned on both the input conversation and dimension d as well.

To ascertain the absolute quality of the text, we pose direct queries to the model using phrases like, "Question: Was the last response from the Assistant helpful?" Conversely, when seeking to determine the relative quality of paired responses, we then prompt the model to rank the two by phrases such as "Rank OPTION A and OPTION B based on two helpful they are".

A notable limitation of encoder reward models is their difficulty in anchoring a model to a specific dimension. For each dimension, there's a need to train a distinct model (Touvron et al 2023), which can be resource-intensive. Amazon GPT-J tries to address this grounding issue with conditional prompt that conditioned on the preference dimension (Reward Modeling). Inspired by this, we also include dimension into our prompt. However, one key difference in our approach is that, instead of simply prepending single dimension token such as “helpful” or “quality” before conversation like Amazon GPT-J, we verbalize the dimension into definition such that decoder model can better parse, interpret, and reason according to the provided definition.

```
In conversation, a helpful Assistant will offer factual knowledge, ask relevant follow-up questions, p
{context}<endofcontext>
Assistant: {response}
Question: Is the last response from Assistant helpful?
Answer:
```

```
In conversation, a helpful Assistant will offer factual knowledge, ask relevant follow-up questions, p
{context}<endofcontext>
OPTION A: Assistant: {response1}
OPTION B: Assistant: {response2}
Question: Rank OPTION A and OPTION B based on how helpful they are.
Answer:
```

In above columns we show prompt examples of helpful dimension that assess absolute quality (Evaluation of absolute quality) and relative quality (Evaluation of relative quality) respectively. The `{context}` is a placeholder to be substituted by actual context during runtime. `{response}, {response1}, {reponse2}` are also similar placeholders.
For more examples, please refer to Decoder reward model prompt design.

Training prompt design

We adapt Chain of Hindsight alignment, designed to align decoder models with various feedback forms. Specifically, we present both high-quality and low-quality responses alongside their context as a singular training instance for the language model. By assimilating diverse feedback from human preference datasets implicitly (i.e. without the direct use of RLHF), the model can explicitly ascertain human preferences. Our hypothesis posits that the implicit amalgamation of both positive and negative feedback augments the reward model's proficiency in discerning human preferences.

Image taken from Chain of Hindsight paper. Here model sees both response A and response B within one context.
Furthermore, one additional bonus of Chain of Hindsight training is Controllable Generation.

We provide one example for helpful dimension:

```
{context}<endofcontext>

A helpful response: Assistant: {response}

An unhelpful response: Assistant: {rejected}
```

Lastly, in order to familiarize models with test-time prompts and instruct the model to output we want it to output, for each of the training datasets we train its Evaluation prompt design with Preference Dimension for one epoch. For all training instances, we apply loss on the generated part only (Vicuna 2023, Taori et al 2023).
For more examples, please refer to Decoder reward model prompt design.

4.2 Model Selection

What are the correct decoder architecture and optimal parameter size?

In an era where Large Language Models (LLMs) are flourishing, with new models emerging weekly, determining a universally preferable model remains an active area of research. Furthermore, a model's exemplary performance on specific academic benchmarks doesn't necessarily guarantee its efficacy in reward modeling. Indeed Fu et al 2023 showed that different models are good on different aspects of the preferences. Lastly, as model scaling leads to emergence ability (Wei et al 2022, Chung et al 2022), model parameter size is also an important factor in measuring the model performance as large models generally indicates better machine understanding capacity.

With the objective of constructing a reward model to support our internal SFT model development, our primary emphasis is on two key aspects: (1) perform sufficiently well on helpful dimension as this is one of the most important preference dimensions (2) the model should not only excel on academic benchmarks but also demonstrate adaptability to our internal conversations, which may present considerable domain disparities. (I made a separate quip doc in [OpenRewardModel] 0628 ADS LLM - Human Eval that includes the full results).

- We collected RAM datasets that consists of actual single-/multi-turn conversations between humans and AGM model. Two human annotators made two independent passes on each of the conversations, and labeled either helpful, somewhat helpful, or unhelpful. We

keep only ones that both annotators agree with helpful or unhelpful as the final datasets, which has 1,826 conversations.

- We select three model architectures with varies model size: OpenLLaMA 7B and 13B, Meta's LLaMA 7B, 13B and 65B, as well as Falcon 7B and 40B. Those models are the models that achieves the best performance across the academic benchmarks by June 2023. It should be also noted that these models have license of free use under research purpose. We pick those base pretrained model rather than a SFT model also due to legal concerns.
- We conduct a zero-shot testing using prompt designed in Evaluation prompt design with Preference Dimension without any in-context demonstrations and the results are shown in the following table. We report F1 for both helpful and unhelpful predictions as the datasets is overly imbalanced towards helpful conversations.

Model	F1 (helpful)	F1 (unhelpful)
OpenLLaMA 7B	0.83	0.49
OpenLLaMA 13B	0.82	0.59
Falcon 7B	0.79	0.47
Falcon 40B	0.88	0.6
LLaMA 7B	0.8	0.61
LLaMA 13B	0.84	0.59
LLaMA 65B	**0.89**	**0.66**

We observe that

1. Among the pretrained models of same arch of different parameter sizes (OpenLLaMA, Meta LLaMA, Falcon), **larger model size is strictly better in both metrics.** Thus parameter size plays a role here, and reward modeling seems to be one of the emergent abilities that only appears when the model is scale up.
2. Among different architectures, **Meta's LLaMA 65B is the best.**

Consequently, we limit our project to utilizing the LLaMA architecture (and the upgraded LLaMA 2) with parameter size greater than or equal to 13B as smaller model does not provide enough performance. Maintaining a focus can save us compute and time cost.

Baselines and Implementation Details

We focus on two models specifically: LLaMA 2 13B and LLaMA 30B. We opt for LLaMA 2 13B LLaMA 13B due to performance supremacy of LLaMA 2 as well as improved efficiency via grouped-query attention (Touvron et al 2023).
Since LLaMA 2 doesn't offer a 30B variant, we turn to the original LLaMA version.
We refrain from venturing beyond the 30B model to strike a balance between the prohibitively high inference costs associated with the larger 60B model and the incremental performance benefits derived from model scaling.

We compare our models against 4 widely used SOTA baselines:

- OpenAssistant DeBERTa v3 (Köpf et al 2023) is a 300M encoder model
- Standford SHP Flan-T5 is 3B (Ethayarajh et al 2022) encoder-decoder model. The scores are calculated using the ratio between logprobs.
- HelpfulRM (Reward Modeling) is Amazon internal GPT-J model that replaces the final language model head with a linear head. HelpfulRM is trained on similar data mixture as our models thus is a better comparsion
- MultidimRM (Reward Modeling) is a variant of HelpfulRM that is trained not only on public datasets but internal ADS data as well. Therefore comparing with MultidimRM is not apple-to-apple as MultidimRM is exposed to more in-domaon training dataset. However as MultidimRM is widely-used internally among SFT team, we believe it is interesting to compare against with.

To the best of our knowledge, **our decoder reward model is the first one to explore using decoder-only architecture itself for reward modeling without any modification on the architecture.**

We train our decoder models on several p4d EC2 instances where inter-instance communication is supported via infiniband. The sequence length is 2048, and the model is trained with 2 epochs with cosine learning rate scheduler. To improve efficiency, Deepspeed ZERO-3 parallelism and CPU offloading is deployed. We also use FlashAttention v2 (Dao et al 2023) to improve throughput. For 13B model we use 8 p4d instances and takes 16 hours to finish. For 30B model we use 16 p4d instances and takes 53 hours to finish.

4.2 Experiments

Evaluation of relative quality

Following previous works (Ganguli et al 2022, Askell et al 2021, Ethayarajh et al 2022, Köpf et al 2023), we first conduct pairwise evaluation where the model is given a pair of inputs and is tasked to determine which one is more preferable given the preference dimension. In traditional encoder reward model, this is computed via calculating scores for two inputs separately and compare the scores. For decoder reward model, we can simply incorporate both inputs into the prompt (Evaluation prompt design with Preference Dimension) and directly instruct the model to output the ranking. Therefore, even though decoder reward models possess a larger parameter size, we empirically observe that the inference cost remains comparable. This is attributed to two factors: (1) Our primary requirement is the ranking, eliminating the need to generate extraneous tokens beyond (2) The decoder necessitates only a single forward pass, as opposed to the encoder's two.
The performance is reported in Fig 1a below with detail numbers in the table below. We report accuracy for all datasets.
Figure 1a, performance of decoder reward model and baselines on relative quality assessment

Figure 1b, performance of decoder reward model and baselines on absolute quality assessment

Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair**					SMF (r summ
	helpful-reject	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
OA-DeBERTa	69.494	67.519	63.617	71.496	51.051	64.255	43.716	67.208	67.707	72.816
SHP-T5	68.198	64.015	61.915	63.196	58.859	64.54	46.995	64.845	65.366	51.092
GPT-J HelpfulRM	63.958	62.405	59.149	**71.751**	53.904	61.505	44.809	64.697	62.439	64.8
GPT-J MultiDimRM	70.789	67.33	59.574	67.674	54.054	63.965	44.08	62.629	64.39	64.219
LLaMA 2 13B	69.38	66.1	**64.47**	66.69	**59.01**	64.54	54.64	**70.9**	70.24	71.14
LLaMA 30B	**71.61**	**67.52**	62.77	69.38	58.86	**66.28**	**57.01**	69.87	**72.59**	**73.54**

We note that

- Both decoder reward model consistently outperforms SOTA baselines across the wide array of datasets except WebGPT. This reinforces our assumption that training decoder reward model directly is feasible
- Decoder reward model is particularly good in Ethics, which is a rather complicated preference dimension that requires commonsense knowledge and multi-hop reasoning. The performance gains possibly should be attributed to reasoning capacity of decoder models.
- Decoder reward model is trained using publicly-available datasets only, yet it still is able to outperform Amazon GPT-J model even if it is trained on internal datasets which are larger-scale and higher-quality.

Evaluation of absolute quality

In certain scenarios, the relative quality of a response is not a primary concern; rather, its absolute quality is more of interest. For instance, during processes like rejection sampling (Askell et al 2021) or data filtering (Köpf et al 2023), the objective is to ascertain if a singular response is helpful.
Motivated by such use case, we also conduct experiments to evaluate the response's absolute quality. For each instance which is a triplet (For fair comparison between baselines (Baselines and Implementation Details), all datasets used in our project is of Pairwise type: each...), we query the model twice, one with `context` + `chosen` , one with `context` + `rejected`. The aim is to determine if the model can consistently assign a positive quality to the former and a negative quality to the latter. We present accuracy metrics for all datasets.

It's worth noting that since encoder reward model baselines yield numerical scores, we employ the natural thresholds specified in their respective papers to categorize the predictions:

- OA DeBERTa uses 0 threshold
- SHP-T5 uses 0.7 threshold
- Amazon GPT-J lacks a predefined threshold, and empirically, we find the model tends to assign similar scores across instances. As a workaround, we slightly deviate from the standard approach by leveraging oracle ground truth: we take 10% of the test data as validation data, compute the average scores for helpful and unhelpful responses on this set, and then use the mean of these two averages as the threshold to differentiate between helpful and unhelpful responses. Therefore we emphasize that scores for HelpfulRM and MultidimRM represent an upper bound since we utilize oracle label information, and comparing directly with these two models is not fair.

We report the results in the Fig 1b above and detailed numbers in the table below.

Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair**					SMF (r summ
	helpful-base	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
OA-DeBERTa	58.657	56.25	55.106	59.301	53.228	55.065	48.816	56.869	57.561	59.631
SHP-T5	59.953	55.161	54.681	57.19	53.604	56.151	50.273	55.317	53.659	50.046
GPT-J HelpfulRM	55.948	55.492	53.936	**63.142**	50.375	57.598	45.537	57.386	56.537	57.179
GPT-J MultiDimRM	58.245	53.119	53.298	58.045	53.078	56.874	46.539	56.795	56.098	56.97
LLaMA 2 13B	63.13	58.71	**60.53**	58.63	**55.11**	56.37	54.28	62.78	61.07	62.06
LLaMA 30B	**65.96**	**60.27**	57.02	60.85	**55.11**	**60.13**	**55.19**	**64.18**	**62.63**	**62.7**

We observe that

- A similar trend as evaluating relative quality — Both decoder reward model consistently outperforms SOTA baselines across the wide array of datasets except WebGPT.
- Even though the family of Amazon GPT-J models leverage oracle label information, decoder reward model is able to outperform these two models in all settings

Decoder enables In-Context Learning

Decoder models are known to learn from demonstrations provided within the prompt context (Brown et al 2020, Min et al 2022, Wei et al 2022). Naturally, decoder reward model will also benefits from additional conversations demonstrations. Such capability is not possible for encoder reward model baselines.
To leverage model's ability to learn from in-context demonstration, we sampled a fix 2 shot demonstration from the training set for each of the datasets.

In the figure 1a and 1b above, we showed the performance boost of using demonstrations, **despite rudimentary, already yield significant boost over zero-shot evaluation on both absolute and relative quality assessment**. It is possible to yield further performance gains by incorporating fancier in-context learning tricks such as self-consistency (Wang et al 2022) or tree of thoughts (Yao et al 2023), which we leave as future direction.

Zero-shot Generalization

Lastly, benefiting from the LLM's inherent reasoning capabilities and robust generalization, the **decoder reward model can also adapt to unseen preference dimension or datasets**. We have conducted experiments on the following 14 internal datasets:

Test Dataset	Subset	Dimension	Type	#instances
ADS	ies	helpful	Pairwise	4323
	interactive-long	helpful	Pairwise	5008
	interactive-short	helpful	Pairwise	10020
	v1-all	helpful	Pairwise	8428
Convs Pairs		helpful	Pairwise	2428
GPT-3	content manipulation	helpful	Pairwise	253
	explanation multi-step	helpful	Pairwise	222
	inaccurate information	accurate	Pairwise	43
	knowledge advanced	helpful	Pairwise	414
	knolwedge world	accurate	Pairwise	414
	less informative	informative	Pairwise	43
	no user following	instruction following	Pairwise	43
	not well formatted	well-formatted	Pairwise	43
	off topic	coherent	Pairwise	43

We first show that decoder reward model can effectively generalize to out-of-domain datasets. Given that the decoder reward model is trained on public datasets, there exists a domain gap between the test datasets and the training datasets. This observation is further corroborated by preliminary eperiments, where models that excel on academic benchmarks do not necessarily perform as well on internal datasets. Nevertheless, as illustrated in the table below, decoder reward model, through its chain-of-hindsight training dataset, is adept at grasping actual preferences. Consequently, it generalizes effectively across 8 out-of-domain datasets.

Relative Quality								
Backbone	**ADS (helpful)**				**convs pairs (helpful)**	**GPT-3**		
	ies	**interactive-long**	**interactive-short**	**v1-all**		**context manipulation (helpful)**	**explanation (helpful)**	**knowledge advanced (helpful)**
OA-DeBERTa	60.213	60.403	56.188	66.338	67.339	87.747	86.486	89.13
SHP-T5	59.889	61.362	55.98	67.489	62.191	88.33	86.486	86.957
GPT-J HelpfulRM	56.095	56.31	54.142	49.312	57.949	77.075	79.279	78.986
GPT-J MultiDimRM	60.861	**61.482**	56.627	68.166	61.944	86.166	89.64	87.44
LLaMA 2 13B	60.75	59.56	56.65	64.42	66.6	**88.54**	89.19	93
LLaMA 30B	**61.92**	60.86	**58.78**	**67.92**	**67.34**	**88.54**	**95.05**	**93.96**
Absolute Quality								
Backbone	**ADS (helpful)**				**convs pairs (helpful)**	**GPT-3**		
	ies	**interactive-long**	**interactive-short**	**v1-all**		**context manipulation (helpful)**	**explanation (helpful)**	**knowledge advanced (helpful)**
OA-DeBERTa	52.822	52.676	50.933	58.003	53.11	75.889	73.198	69.324
SHP-T5	52.232	53.235	51.337	58.602	52.657	69.96	76.802	72.222
GPT-J HelpfulRM	51.897	52.945	50.654	49.549	53.418	65.217	75.901	71.981
GPT-J MultiDimRM	52.255	52.446	50.704	57.517	53.171	58.696	63.514	59.783
LLaMA 2 13B	55.75	54.83	53.96	55.47	**60.42**	75.1	74.5	70.29
LLaMA 30B	**56.27**	**55.14**	**54.94**	**59.48**	60.28	**78.26**	**82.66**	**78.99**

Secondly, and perhaps more interestingly, we show that decoder reward model can generalize to unseen preference dimensions as well. In practical scenarios, there might be a need for more nuanced dimensions as opposed to broader ones like "helpful." For instance, the "instruction

following" dimension could arguably be more critical than "helpful" for instruction-tuning models. However, given that preference datasets are designed to emulate human preferences, there's an inherent necessity to gather datasets from human annotations for each of these finer-grained preference dimensions, which can be prohibitively expensive.
The table below indicates that even though the decoder reward model hasn't been trained on or exposed to these dimensions previously, it can still deliver satisfactory performance. This is achieved by parsing the verbalized definition of the target preference dimension and subsequently reasoning about the response's quality in relation to that specific dimension.

Relative Quality						
Backbone	**GPT-3**					
	inaccurate (accurate)	**knowledge world (accurate)**	**less informative (informative)**	**no user following (instruct follow)**	**not well (well formatted)**	**off topic (coherent)**
OA-DeBERTa	93.02	91.383	**100**	88.372	88.372	**100**
SHP-T5	62.791	87.982	95.349	72.093	68.767	**100**
GPT-J HelpfulRM	58.14	78.912	95.349	81.395	79.07	88.373
GPT-J MultiDimRM	58.14	82.993	97.674	76.744	53.488	95.349
LLaMA 2 13B	93.02	93.65	**100**	**90.7**	93.02	**100**
LLaMA 30B	**95.35**	**94.78**	**100**	83.72	**95.35**	**100**
Absolute Quality						
Backbone	**GPT-3**					
	inaccurate (accurate)	**knowledge world (accurate)**	**less informative (informative)**	**no user following (instruct follow)**	**not well (well formatted)**	**off topic (coherent)**
OA-DeBERTa	77.907	83.9	90.698	70.93	69.767	95.349
SHP-T5	61.628	**77.438**	82.558	68.605	60.465	**100**
GPT-J HelpfulRM	51.163	70.181	80.233	70.93	53.488	75.581
GPT-J MultiDimRM	52.326	61.791	67.442	62.791	59.302	75.581
LLaMA 2 13B	84.88	74.04	90.7	**79.07**	**86.05**	91.86
LLaMA 30B	**86.05**	76.76	**94.19**	77.91	80.23	90.7

Robust to prompt design

Some critics might contend that the prompt designs from Evaluation prompt design with Preference Dimension are somewhat arbitrary or selectively chosen. To address this, we modified the definition of each preference dimension. We provided the original definition to ChatGPT and requested a rewrite to diversify the expression. We set the temperature to 0.7 to infuse greater randomness and variety.
The table below presents the results of the 30B decoder reward model using these newly rewritten prompts. Notably, even when the definitions differ from what the model encountered during training, it generally continues to deliver commendable performance. In some instances, it even surpasses the results achieved with the original prompts. This observation suggests that the **model isn't merely relying on rote memorization of the definitions but possesses an intrinsic reasoning capability to interpret and act upon the provided definitions.**

Relative Quality										
Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair** (Accuracy)					SMF (1 sum
	helpful-base	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
LLaMA 30B	68.67	**66.86**	**65.32**	**64.58**	56.31	63.24	47.91	65.14	65.07	**68.01**
w/ new prompt	**69.6**	62.4	57.4	61.6	**56.2**	**65.6**	**54.6**	**66.2**	**65.5**	67.9
Absolute Quality										
Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair** (Accuracy)					SMF (1 sum
	helpful-base	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
LLaMA 30B	**64.96**	60.13	59.26	59.14	**58.18**	60.06	55.37	63.07	61.41	**63.06**

w/ new prompt	64.7	**61.3**	**61.3**	**59.4**	57.2	**60.5**	**55.7**	**64.3**	**61.6**	62.4

Decoder Reward Model is on par with propriety LLMs

The aforementioned results offer a comparison with state-of-the-art public models trained on widely available datasets. However, it's becoming increasingly common for researchers to employ proprietary LLMs for evaluating conversation quality. To gauge the performance of the decoder reward model against these proprietary LLMs, we selected 100 instances from each of the 11 datasets listed below, and computed both relative and absolute quality evaluations.
We benchmarked against three prevalent LLMs: Claude 2 from Anthropic, and both ChatGPT and GPT4 from OpenAI. To ensure a balanced comparison, we supplied the same prompts used by the decoder reward model to these LLMs.
Indeed, our observations confirm that GPT4 excels in evaluating conversation quality, particularly in terms of relative assessment. Yet, our **decoder reward model is also capable of producing results that are on par, closely rivaling LLMs that may be up to 100 times larger than the 30B model. Notably, even if our model doesn't clinch the top spot, it consistently ranks within the top two, outperforming ChatGPT in most scenarios.**

Relative Quality										
Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair**					SMF (r summ
	helpful-base	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
Claude 2	60	54	53	63	63	69	59	69	66	64
ChatGPT	59	56	52	73	64	62	60	63	**67**	64
GPT4	62	60	**63**	77	**66**	**70**	**64**	**73**	66	**80**
LLaMA 2 13b	**65**	53	54	59	55	58	61	61	65	57
LLaMA 1 30b	**65**	**62**	61	70	**66**	69	49	70	66	75
Absolute Quality										
Backbone	**Anthropic-helpful** (helpful)			**WebGPT** (helpful)	**Open Assistant Pair**					SMF (r summ
	helpful-base	**helpful-reject**	**helpful-online**		**Creative**	**Helpful**	**Humor**	**Quality**	**Rank (Quality)**	
Claude 2	51.5	53.5	51.5	50	51	53	44	48	49	53
ChatGPT	54.5	50.5	51.5	51	55.5	55	**58.5**	55	52	52
GPT4	58	59	54.5	57	**57.5**	54	55	58.5	57	54
LLaMA 2 13b	58.5	60	53	**63**	56	58	55	59	64.5	61
LLaMA 1 30b	**60.5**	**60.5**	**57.5**	59.5	54.5	**59.5**	50.5	**64**	**66.5**	**63**

5. Beyond Reward Modeling

In 4. Reward Modeling we showed that our decoder reward model is able to achieve consistent improvement over results with public SOTA, and can achieve comparable results even with propriety LLMs that are much larger and more expensive. However we do not stop merely at here. We question: **what can a LLM that learns to discern quality response do to provide further benefits?**

5. 1 Reward Modeling Filtering

In corporation like Amazon, it is often possible to collect large scale datasets. However a majority of the dataset can be noisy (Veselovsky et al 2023) thus not suitable to use for SFT as noisy instances can lead to worse performance for LLM (Wei et al 2023). One potential way to resolve this issue is to reduce the seize via reward model scores. Such filtered dataset might lead to better performance despite dramatically smaller.
We compare decoder RM scores against baselines on two internal datasets.
For each of the datasets, we first filter by top-k reward model scores, then within these top-k instances, we further downsample via k-mean clustering on the sentence embedding of the conversations. The rationale is that, according to LIMA (Zhou et al 2023), we need both high quality instances as well as diverse instances to make a strong SFT models. Since top-k scores can only guarantee quality, we use kmean clustered instances as one proxy for diversity. We use sentence transformers' all-mpnet-base-v2 to embedding the entire conversations (all turns, including assistant's response).
To validate the effectiveness of the filtering, we train AGMv2 model (`agmv2-7b_1166b-2k`) on top of the filtered datasets and evaluate the model performance against models trained on the entire unfiltered dataset. We use Amazon GPT-J for evaluation since this is what used internally among SFT team. Higher is better.

- We filter Bedrock single-turn datasets from 382k to 50k, reducing to only 13% of the size. We explored two different configurations: using only top-50k or using top-100k, then clustering those top-100k instances into 500 clusters. Within each cluster we pick 100 points.

Model	Helpful	Quality	Instruction Following	Engaging	Interesting	Relevant	Coherent
drm+kmeans	**-11.96953**	**-11.2467**	**-11.36916**	**-11.66252**	**-11.49067**	**-11.73208**	**-11.59529**
oa+kmeans	-12.09827	-11.27438	-11.4157	-12.31727	-11.62688	-11.68445	-11.71877
drm	-12.37033	-11.59101	-11.86163	-12.38173	-11.91964	-11.95847	-11.88791
gptj	-12.53209	-13.03994	-13.13858	-13.69227	-13.11567	-13.11164	-13.2516
gptj+kmeans	-12.68414	-13.41141	-13.39543	-14.10667	-13.32084	-13.44052	-13.44931
shp+kmeans	-12.68966	-13.859	-13.68324	-13.98634	-13.83762	-13.81049	-13.91989

shp	-12.74072	-13.0896	-13.1204	-13.86579	-13.08568	-13.27976	-13.47186
oa	-12.78221	-12.30972	-12.77326	-13.79549	-12.60515	-12.63358	-12.53977
original	-12.81733	-13.60634	-13.64254	-14.32352	-13.62332	-13.66799	-13.85959
random	-13.04263	-13.52238	-13.79665	-14.65016	-13.67664	-13.50081	-13.81004

We observe that kmeans generally outperforms top-k filtering. But any of the filtering outperforms traing on the entire bedrock which can be noisy. On the other hand, random 50k is the worse, possibly due to high chance of random samples being noisy.
Our results reinforcement our assumption that we need both quality (top-100k) and diversity (500 clusters) to achieve good SFT models. We also note that decoder reward model performs the best for both top-100k only and top-100k+clustering configurations.

5.2 Rationale Generation

Experiments on out-of-domain generalization (Zero-shot Generalization) indicates that domain gap does exists among preference datasets. In other words, we humans have alignment issues too — my definition of helpful might differ from yours.
One illuminative example for this "alignment drift" problem is given below:
Human: What is the net worth of julie sweet?
Assistant: According to Forbes, Julie Sweet has a net worth of $20 million. She has earned her wealth primarily through her work as an actress and fashion model.
Human: What is the net worth of julie sweet?
Assistant: I'm not able to provide information on the net worth of individuals as this information is not publicly disclosed and could be considered private…
Left is an actual conversation in the Anthropic dataset. Despite the query seeking information about a public figure, the response is labeled as a helpful reply. Conversely on the right is my personal conversation with LLaMA 2 70B chat, who refuses to provide personal information. This leads me to surmise that LLaMA 2 was trained on a dataset that categorizes such queries as unhelpful, in stark contrast to Anthropic's HHH definition of helpfulness.

However all existing pairwise datasets contain only the chosen response or rejected response, without providing an explanation of why behind a particular decision. We believe that an ideal preference dataset should provide explanation to avoid confusion such as above conversations. Blindly feeding both contradicting instances to the model would only confuse the model and downgrade the performance (Wei et al 2023). Yet, conducting human annotation for each of the training instances can be prohibitively expensive. For example, Anthropic HHH contains over 100k instances. It would cost thousands of dollars to recruit people annotate each of the 100k instances.

One benefits of decoder reward model is that, by simple prompting ("provide reasoning and think step by step"), it is possible to elicit rationale behind model's prediction. On Anthropic HHH, we apply such prompt on each of the training instances and produce 106k rationales. To ensure that such generated rationale aligns with human reasoning, we conduct human evaluation to testify whether this generated rationale is of higher quality.
To save cost, we sample 100 instances from generated 106k rationales and ask annotators to rate the quality of the rationale. Eventually we will compare the ratio of high-quality rationales to see whether decoder reward modal is as better than ChatGPT.

[waiting for human eval to be done]

5.3 AI Self-refinement

Recent witness a heated discussion on AI improving AI itself (Madaan et al 2023, Yang et al 2022, Huang et al 2022). Specifically, LLM can train on the synthetic dataset produced by the LLM itself, and once the LLM is trained/improved it can create more high-quality synthetic dataset and continue the cycle.
Motivated by these previous works, we question whether decoder reward model can actively improve itself through self-refinement.

Controllable Generation

As decoder reward model is training with Chain of Hindsight training (Training prompt design), the model is able to controlled generate response for the targeted preference dimension.
For example, to generate a helpful response, we can simply prompting the model via

```
{context}
A more helpful response: Assistant:
```

since the model is trained to predict the helpful response.
On the other hand, to create less helpful response, it is easy as to prompt

```
{context}
A less helpful response: Assistant:
```

Complete the cycle

[waiting for new iterations to be done]