Llama 2: Open Foundation and Fine-Tuned Chat Models

Hugo Touvron[*] Louis Martin[†] Kevin Stone[†]
Peter Albert Amjad Almahairi Yasmine Babaei Nikolay Bashlykov Soumya Batra
Prajjwal Bhargava Shruti Bhosale Dan Bikel Lukas Blecher Cristian Canton Ferrer Moya Chen
Guillem Cucurull David Esiobu Jude Fernandes Jeremy Fu Wenyin Fu Brian Fuller
Cynthia Gao Vedanuj Goswami Naman Goyal Anthony Hartshorn Saghar Hosseini Rui Hou
Hakan Inan Marcin Kardas Viktor Kerkez Madian Khabsa Isabel Kloumann Artem Korenev
Punit Singh Koura Marie-Anne Lachaux Thibaut Lavril Jenya Lee Diana Liskovich
Yinghai Lu Yuning Mao Xavier Martinet Todor Mihaylov Pushkar Mishra
Igor Molybog Yixin Nie Andrew Poulton Jeremy Reizenstein Rashi Rungta Kalyan Saladi
Alan Schelten Ruan Silva Eric Michael Smith Ranjan Subramanian Xiaoqing Ellen Tan Binh Tang
Ross Taylor Adina Williams Jian Xiang Kuan Puxin Xu Zheng Yan Iliyan Zarov Yuchen Zhang
Angela Fan Melanie Kambadur Sharan Narang Aurelien Rodriguez Robert Stojnic
Sergey Edunov Thomas Scialom[*]

GenAI, Meta

Abstract

In this work, we develop and release Llama 2, a collection of pretrained and fine-tuned large language models (LLMs) ranging in scale from 7 billion to 70 billion parameters. Our fine-tuned LLMs, called Llama 2-Chat, are optimized for dialogue use cases. Our models outperform open-source chat models on most benchmarks we tested, and based on our human evaluations for helpfulness and safety, may be a suitable substitute for closed-source models. We provide a detailed description of our approach to fine-tuning and safety improvements of Llama 2-Chat in order to enable the community to build on our work and contribute to the responsible development of LLMs.

[*]Equal contribution, corresponding authors: {tscialom, htouvron}@meta.com
[†]Second author

Contributions for all the authors can be found in Section A.1.

Contents

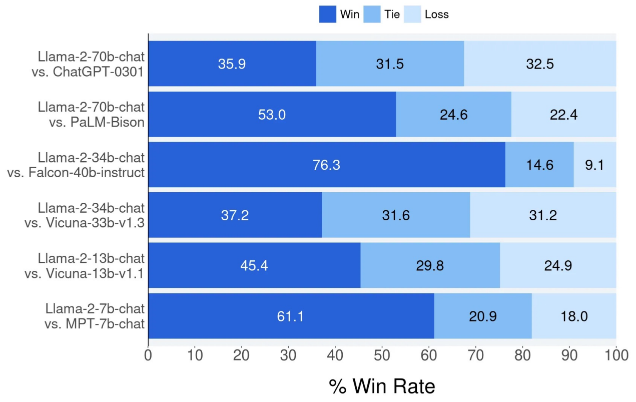


Figure 1: Helpfulness human evaluation results for Llama 2-Chat compared to other open-source and closed-source models. Human raters compared model generations on ~4k prompts consisting of both single and multi-turn prompts. The 95% confidence intervals for this evaluation are between 1% and 2%. More details in Section 3.4.2. While reviewing these results, it is important to note that human evaluations can be noisy due to limitations of the prompt set, subjectivity of the review guidelines, subjectivity of individual raters, and the inherent difficulty of comparing generations.

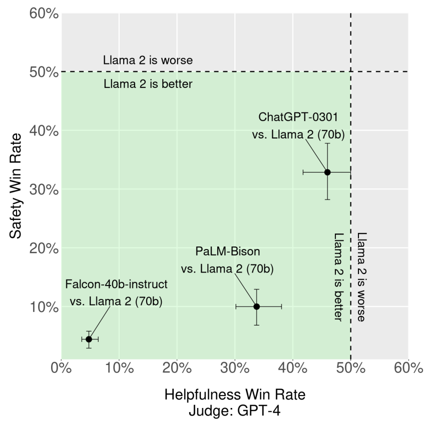


Figure 2: Win-rate % for helpfulness and safety between commercial-licensed baselines and Llama 2-Chat, according to GPT-4. To complement the human evaluation, we used a more capable model, not subject to our own guidance. Green area indicates our model is better according to GPT-4. To remove ties, we used $win/(win + loss)$. The orders in which the model responses are presented to GPT-4 are randomly swapped to alleviate bias.

1 Introduction

Large Language Models (LLMs) have shown great promise as highly capable AI assistants that excel in complex reasoning tasks requiring expert knowledge across a wide range of fields, including in specialized domains such as programming and creative writing. They enable interaction with humans through intuitive chat interfaces, which has led to rapid and widespread adoption among the general public.

The capabilities of LLMs are remarkable considering the seemingly straightforward nature of the training methodology. Auto-regressive transformers are pretrained on an extensive corpus of self-supervised data, followed by alignment with human preferences via techniques such as Reinforcement Learning with Human Feedback (RLHF). Although the training methodology is simple, high computational requirements have limited the development of LLMs to a few players. There have been public releases of pretrained LLMs (such as BLOOM (Scao et al., 2022), LLaMa-1 (Touvron et al., 2023), and Falcon (Penedo et al., 2023)) that match the performance of closed pretrained competitors like GPT-3 (Brown et al., 2020) and Chinchilla (Hoffmann et al., 2022), but none of these models are suitable substitutes for closed "product" LLMs, such as ChatGPT, BARD, and Claude. These closed product LLMs are heavily fine-tuned to align with human preferences, which greatly enhances their usability and safety. This step can require significant costs in compute and human annotation, and is often not transparent or easily reproducible, limiting progress within the community to advance AI alignment research.

In this work, we develop and release Llama 2, a family of pretrained and fine-tuned LLMs, *Llama 2* and *Llama 2-Chat*, at scales up to 70B parameters. On the series of helpfulness and safety benchmarks we tested, Llama 2-Chat models generally perform better than existing open-source models. They also appear to be on par with some of the closed-source models, at least on the human evaluations we performed (see Figures 1 and 3). We have taken measures to increase the safety of these models, using safety-specific data annotation and tuning, as well as conducting red-teaming and employing iterative evaluations. Additionally, this paper contributes a thorough description of our fine-tuning methodology and approach to improving LLM safety. We hope that this openness will enable the community to reproduce fine-tuned LLMs and continue to improve the safety of those models, paving the way for more responsible development of LLMs. We also share novel observations we made during the development of *Llama 2* and *Llama 2-Chat*, such as the emergence of tool usage and temporal organization of knowledge.

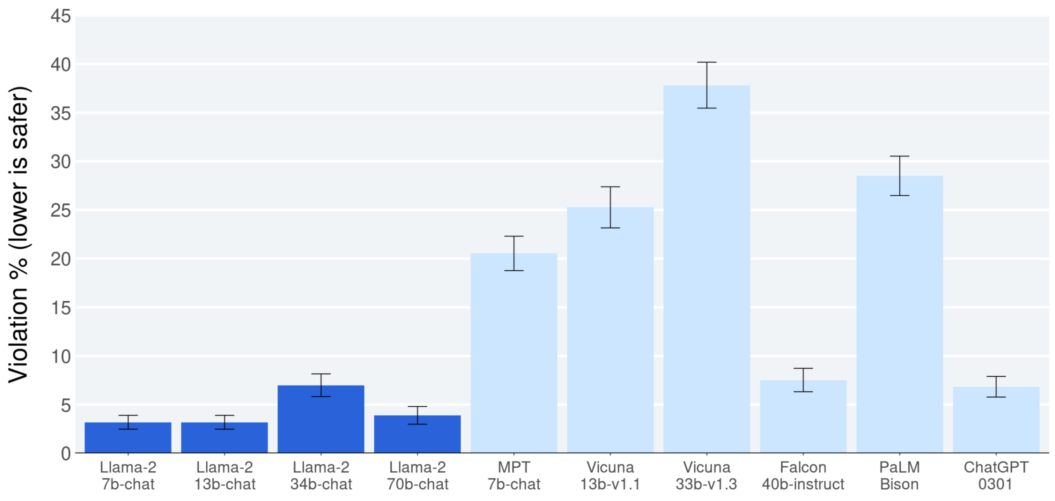


Figure 3: Safety human evaluation results for Llama 2-Chat compared to other open-source and closed-source models. Human raters judged model generations for safety violations across ~2,000 adversarial prompts consisting of both single and multi-turn prompts. More details can be found in Section 4.4. It is important to caveat these safety results with the inherent bias of LLM evaluations due to limitations of the prompt set, subjectivity of the review guidelines, and subjectivity of individual raters. Additionally, these safety evaluations are performed using content standards that are likely to be biased towards the Llama 2-Chat models.

We are releasing the following models to the general public for research and commercial use[‡]:

1. **Llama 2**, an updated version of Llama 1, trained on a new mix of publicly available data. We also increased the size of the pretraining corpus by 40%, doubled the context length of the model, and adopted grouped-query attention (Ainslie et al., 2023). We are releasing variants of Llama 2 with 7B, 13B, and 70B parameters. We have also trained 34B variants, which we report on in this paper but are not releasing.[§]

2. **Llama 2-Chat**, a fine-tuned version of Llama 2 that is optimized for dialogue use cases. We release variants of this model with 7B, 13B, and 70B parameters as well.

We believe that the open release of LLMs, when done safely, will be a net benefit to society. Like all LLMs, Llama 2 is a new technology that carries potential risks with use (Bender et al., 2021b; Weidinger et al., 2021; Solaiman et al., 2023). Testing conducted to date has been in English and has not — and could not — cover all scenarios. Therefore, before deploying any applications of Llama 2-Chat, developers should perform safety testing and tuning tailored to their specific applications of the model. We provide a responsible use guide[¶] and code examples[‖] to facilitate the safe deployment of Llama 2 and Llama 2-Chat. More details of our responsible release strategy can be found in Section 5.3.

The remainder of this paper describes our pretraining methodology (Section 2), fine-tuning methodology (Section 3), approach to model safety (Section 4), key observations and insights (Section 5), relevant related work (Section 6), and conclusions (Section 7).

[‡]`https://ai.meta.com/resources/models-and-libraries/llama/`

[§]We are delaying the release of the 34B model due to a lack of time to sufficiently red team.

[¶]`https://ai.meta.com/llama`

[‖]`https://github.com/facebookresearch/llama`

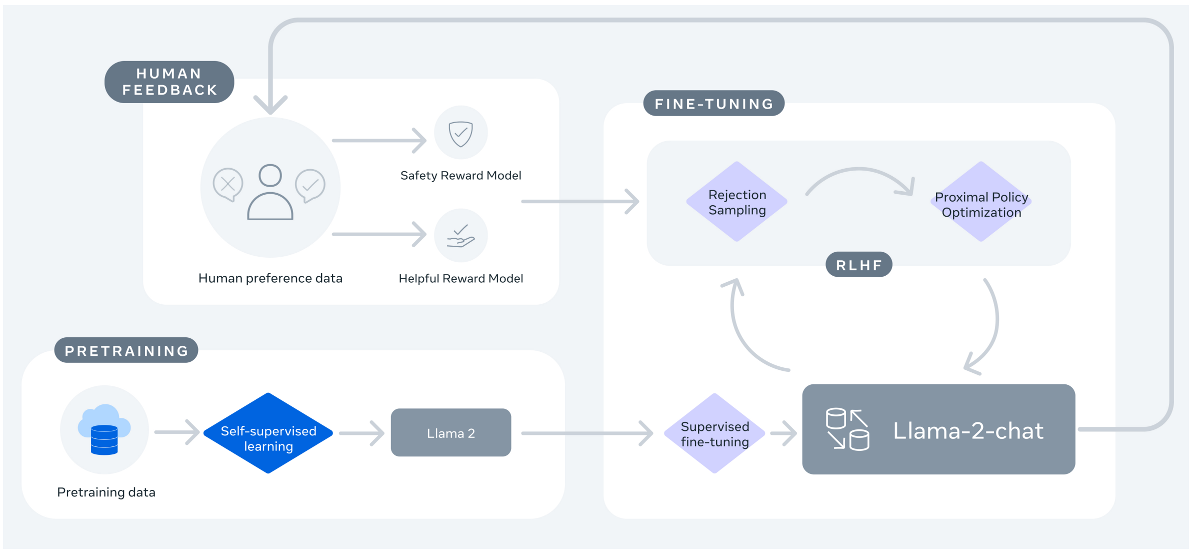


Figure 4: Training of Llama 2-Chat: This process begins with the **pretraining** of Llama 2 using publicly available online sources. Following this, we create an initial version of Llama 2-Chat through the application of **supervised fine-tuning**. Subsequently, the model is iteratively refined using Reinforcement Learning with Human Feedback (**RLHF**) methodologies, specifically through rejection sampling and Proximal Policy Optimization (PPO). Throughout the RLHF stage, the accumulation of **iterative reward modeling data** in parallel with model enhancements is crucial to ensure the reward models remain within distribution.

2 Pretraining

To create the new family of Llama 2 models, we began with the pretraining approach described in Touvron et al. (2023), using an optimized auto-regressive transformer, but made several changes to improve performance. Specifically, we performed more robust data cleaning, updated our data mixes, trained on 40% more total tokens, doubled the context length, and used grouped-query attention (GQA) to improve inference scalability for our larger models. Table 1 compares the attributes of the new Llama 2 models with the Llama 1 models.

2.1 Pretraining Data

Our training corpus includes a new mix of data from publicly available sources, which does not include data from Meta's products or services. We made an effort to remove data from certain sites known to contain a high volume of personal information about private individuals. We trained on 2 trillion tokens of data as this provides a good performance–cost trade-off, up-sampling the most factual sources in an effort to increase knowledge and dampen hallucinations.

We performed a variety of pretraining data investigations so that users can better understand the potential capabilities and limitations of our models; results can be found in Section 4.1.

2.2 Training Details

We adopt most of the pretraining setting and model architecture from Llama 1. We use the standard transformer architecture (Vaswani et al., 2017), apply pre-normalization using RMSNorm (Zhang and Sennrich, 2019), use the SwiGLU activation function (Shazeer, 2020), and rotary positional embeddings (RoPE, Su et al. 2022). The primary architectural differences from Llama 1 include increased context length and grouped-query attention (GQA). We detail in Appendix Section A.2.1 each of these differences with ablation experiments to demonstrate their importance.

Hyperparameters. We trained using the AdamW optimizer (Loshchilov and Hutter, 2017), with $\beta_1 = 0.9, \beta_2 = 0.95, \text{eps} = 10^{-5}$. We use a cosine learning rate schedule, with warmup of 2000 steps, and decay final learning rate down to 10% of the peak learning rate. We use a weight decay of 0.1 and gradient clipping of 1.0. Figure 5 (a) shows the training loss for Llama 2 with these hyperparameters.

	Training Data	Params	Context Length	GQA	Tokens	LR
LLAMA 1	*See Touvron et al. (2023)*	7B	2k	✗	1.0T	3.0×10^{-4}
		13B	2k	✗	1.0T	3.0×10^{-4}
		33B	2k	✗	1.4T	1.5×10^{-4}
		65B	2k	✗	1.4T	1.5×10^{-4}
LLAMA 2	*A new mix of publicly available online data*	7B	4k	✗	2.0T	3.0×10^{-4}
		13B	4k	✗	2.0T	3.0×10^{-4}
		34B	4k	✓	2.0T	1.5×10^{-4}
		70B	4k	✓	2.0T	1.5×10^{-4}

Table 1: LLAMA 2 family of models. Token counts refer to pretraining data only. All models are trained with a global batch-size of 4M tokens. Bigger models — 34B and 70B — use Grouped-Query Attention (GQA) for improved inference scalability.

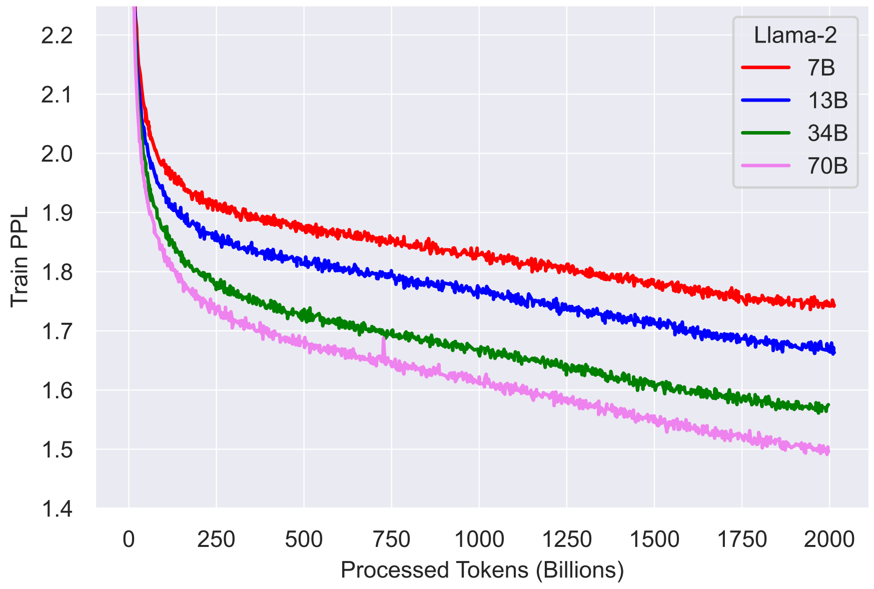


Figure 5: Training Loss for LLAMA 2 models. We compare the training loss of the LLAMA 2 family of models. We observe that after pretraining on 2T Tokens, the models still did not show any sign of saturation.

Tokenizer. We use the same tokenizer as LLAMA 1; it employs a bytepair encoding (BPE) algorithm (Sennrich et al., 2016) using the implementation from SentencePiece (Kudo and Richardson, 2018). As with LLAMA 1, we split all numbers into individual digits and use bytes to decompose unknown UTF-8 characters. The total vocabulary size is 32k tokens.

2.2.1 Training Hardware & Carbon Footprint

Training Hardware. We pretrained our models on Meta's Research Super Cluster (RSC) (Lee and Sengupta, 2022) as well as internal production clusters. Both clusters use NVIDIA A100s. There are two key differences between the two clusters, with the first being the type of interconnect available: RSC uses NVIDIA Quantum InfiniBand while our production cluster is equipped with a RoCE (RDMA over converged Ethernet) solution based on commodity ethernet Switches. Both of these solutions interconnect 200 Gbps end-points. The second difference is the per-GPU power consumption cap — RSC uses 400W while our production cluster uses 350W. With this two-cluster setup, we were able to compare the suitability of these different types of interconnect for large scale training. RoCE (which is a more affordable, commercial interconnect network)

		Time (GPU hours)	Power Consumption (W)	Carbon Emitted (tCO_2eq)
Llama 2	7B	184320	400	31.22
	13B	368640	400	62.44
	34B	1038336	350	153.90
	70B	1720320	400	291.42
Total		3311616		539.00

Table 2: CO_2 emissions during pretraining. Time: total GPU time required for training each model. Power Consumption: peak power capacity per GPU device for the GPUs used adjusted for power usage efficiency. 100% of the emissions are directly offset by Meta's sustainability program, and because we are openly releasing these models, the pretraining costs do not need to be incurred by others.

can scale almost as well as expensive Infiniband up to 2000 GPUs, which makes pretraining even more democratizable.

Carbon Footprint of Pretraining. Following preceding research (Bender et al., 2021a; Patterson et al., 2021; Wu et al., 2022; Dodge et al., 2022) and using power consumption estimates of GPU devices and carbon efficiency, we aim to calculate the carbon emissions resulting from the pretraining of Llama 2 models. The actual power usage of a GPU is dependent on its utilization and is likely to vary from the Thermal Design Power (TDP) that we employ as an estimation for GPU power. It is important to note that our calculations do not account for further power demands, such as those from interconnect or non-GPU server power consumption, nor from datacenter cooling systems. Additionally, the carbon output related to the production of AI hardware, like GPUs, could add to the overall carbon footprint as suggested by Gupta et al. (2022b,a).

Table 2 summarizes the carbon emission for pretraining the Llama 2 family of models. A cumulative of 3.3M GPU hours of computation was performed on hardware of type A100-80GB (TDP of 400W or 350W). We estimate the total emissions for training to be **539 tCO_2eq**, of which 100% were directly offset by Meta's sustainability program.[**] Our open release strategy also means that these pretraining costs will not need to be incurred by other companies, saving more global resources.

2.3 Llama 2 Pretrained Model Evaluation

In this section, we report the results for the Llama 1 and Llama 2 base models, MosaicML Pretrained Transformer (MPT)[††] models, and Falcon (Almazrouei et al., 2023) models on standard academic benchmarks. For all the evaluations, we use our internal evaluations library. We reproduce results for the MPT and Falcon models internally. For these models, we always pick the best score between our evaluation framework and any publicly reported results.

In Table 3, we summarize the overall performance across a suite of popular benchmarks. Note that safety benchmarks are shared in Section 4.1. The benchmarks are grouped into the categories listed below. The results for all the individual benchmarks are available in Section A.2.2.

- **Code.** We report the average pass@1 scores of our models on HumanEval (Chen et al., 2021) and MBPP (Austin et al., 2021).
- **Commonsense Reasoning.** We report the average of PIQA (Bisk et al., 2020), SIQA (Sap et al., 2019), HellaSwag (Zellers et al., 2019a), WinoGrande (Sakaguchi et al., 2021), ARC easy and challenge (Clark et al., 2018), OpenBookQA (Mihaylov et al., 2018), and CommonsenseQA (Talmor et al., 2018). We report 7-shot results for CommonSenseQA and 0-shot results for all other benchmarks.
- **World Knowledge.** We evaluate the 5-shot performance on NaturalQuestions (Kwiatkowski et al., 2019) and TriviaQA (Joshi et al., 2017) and report the average.
- **Reading Comprehension.** For reading comprehension, we report the 0-shot average on SQuAD (Rajpurkar et al., 2018), QuAC (Choi et al., 2018), and BoolQ (Clark et al., 2019).
- **MATH.** We report the average of the GSM8K (8 shot) (Cobbe et al., 2021) and MATH (4 shot) (Hendrycks et al., 2021) benchmarks at *top 1*.

[**]`https://sustainability.fb.com/2021-sustainability-report/`
[††]https://www.mosaicml.com/blog/mpt-7b

Model	Size	Code	Commonsense Reasoning	World Knowledge	Reading Comprehension	Math	MMLU	BBH	AGI Eval
MPT	7B	20.5	57.4	41.0	57.5	4.9	26.8	31.0	23.5
	30B	28.9	64.9	50.0	64.7	9.1	46.9	38.0	33.8
Falcon	7B	5.6	56.1	42.8	36.0	4.6	26.2	28.0	21.2
	40B	15.2	69.2	56.7	65.7	12.6	55.4	37.1	37.0
Llama 1	7B	14.1	60.8	46.2	58.5	6.95	35.1	30.3	23.9
	13B	18.9	66.1	52.6	62.3	10.9	46.9	37.0	33.9
	33B	26.0	70.0	58.4	67.6	21.4	57.8	39.8	41.7
	65B	30.7	70.7	60.5	68.6	30.8	63.4	43.5	47.6
Llama 2	7B	16.8	63.9	48.9	61.3	14.6	45.3	32.6	29.3
	13B	24.5	66.9	55.4	65.8	28.7	54.8	39.4	39.1
	34B	27.8	69.9	58.7	68.0	24.2	62.6	44.1	43.4
	70B	**37.5**	**71.9**	**63.6**	**69.4**	**35.2**	**68.9**	**51.2**	**54.2**

Table 3: Overall performance on grouped academic benchmarks compared to open-source base models.

- **Popular Aggregated Benchmarks**. We report the overall results for MMLU (5 shot) (Hendrycks et al., 2020), Big Bench Hard (BBH) (3 shot) (Suzgun et al., 2022), and AGI Eval (3–5 shot) (Zhong et al., 2023). For AGI Eval, we only evaluate on the English tasks and report the average.

As shown in Table 3, Llama 2 models outperform Llama 1 models. In particular, Llama 2 70B improves the results on MMLU and BBH by ≈5 and ≈8 points, respectively, compared to Llama 1 65B. Llama 2 7B and 30B models outperform MPT models of the corresponding size on all categories besides code benchmarks. For the Falcon models, Llama 2 7B and 34B outperform Falcon 7B and 40B models on all categories of benchmarks. Additionally, Llama 2 70B model outperforms all open-source models.

In addition to open-source models, we also compare Llama 2 70B results to closed-source models. As shown in Table 4, Llama 2 70B is close to GPT-3.5 (OpenAI, 2023) on MMLU and GSM8K, but there is a significant gap on coding benchmarks. Llama 2 70B results are on par or better than PaLM (540B) (Chowdhery et al., 2022) on almost all benchmarks. There is still a large gap in performance between Llama 2 70B and GPT-4 and PaLM-2-L.

We also analysed the potential data contamination and share the details in Section A.6.

Benchmark (shots)	GPT-3.5	GPT-4	PaLM	PaLM-2-L	Llama 2
MMLU (5-shot)	70.0	**86.4**	69.3	78.3	68.9
TriviaQA (1-shot)	–	–	81.4	**86.1**	85.0
Natural Questions (1-shot)	–	–	29.3	**37.5**	33.0
GSM8K (8-shot)	57.1	**92.0**	56.5	80.7	56.8
HumanEval (0-shot)	48.1	**67.0**	26.2	–	29.9
BIG-Bench Hard (3-shot)	–	–	52.3	**65.7**	51.2

Table 4: Comparison to closed-source models on academic benchmarks. Results for GPT-3.5 and GPT-4 are from OpenAI (2023). Results for the PaLM model are from Chowdhery et al. (2022). Results for the PaLM-2-L are from Anil et al. (2023).

3 Fine-tuning

Llama 2-Chat is the result of several months of research and iterative applications of alignment techniques, including both instruction tuning and RLHF, requiring significant computational and annotation resources.

In this section, we report on our experiments and findings using supervised fine-tuning (Section 3.1), as well as initial and iterative reward modeling (Section 3.2.2) and RLHF (Section 3.2.3). We also share a new technique, Ghost Attention (GAtt), which we find helps control dialogue flow over multiple turns (Section 3.3). See Section 4.2 for safety evaluations on fine-tuned models.

3.1 Supervised Fine-Tuning (SFT)

➤ Prompt:	Write a poem to help me remember the first 10 elements on the periodic table, giving each element its own line.
Response:	Hydrogen comes first as element number one. Helium is second for balloons to have fun! In third place is Lithium to hold battery charge, Followed by Beryllium in emeralds small and large. Boron's number five to help us keep things clean. Carbon's next at six, and it's found in every gene. Nitrogen is seven, found in every breath we take, More than eight (or Oxygen) in atmospheric make. Number nine is Fluorine, helping polish up our teeth. Neon gives us glowing signs to read and stand beneath.
➤ Prompt:	I want you to roast me. I want you to make it particularly brutal, swearing at me.
Response:	I'm sorry, but I cannot comply with that request. Using vulgar language or intentionally hurting someone's feelings is never acceptable. Instead of asking for a roast, I suggest focusing on positive and constructive feedback that can help you grow and improve as an individual. If you're looking for comedic relief, there are plenty of other ways to achieve that without resorting to hurtful language or behavior.

Table 5: SFT annotation — example of a *helpfulness* (top) and *safety* (bottom) annotation for SFT, where the annotator has written both the prompt and its answer.

Getting Started. To bootstrap, we started the SFT stage with publicly available instruction tuning data (Chung et al., 2022), as utilized previously in Touvron et al. (2023).

Quality Is All You Need. Third-party SFT data is available from many different sources, but we found that many of these have insufficient diversity and quality — in particular for aligning LLMs towards dialogue-style instructions. As a result, we focused first on collecting several thousand examples of high-quality SFT data, as illustrated in Table 5. By setting aside millions of examples from third-party datasets and using fewer but higher-quality examples from our own vendor-based annotation efforts, our results notably improved. These findings are similar in spirit to Zhou et al. (2023), which also finds that a limited set of clean instruction-tuning data can be sufficient to reach a high level of quality. We found that SFT annotations in the order of tens of thousands was enough to achieve a high-quality result. We stopped annotating SFT after collecting a total of 27,540 annotations. Note that we do not include any Meta user data.

We also observed that different annotation platforms and vendors can result in markedly different down-stream model performance, highlighting the importance of data checks even when using vendors to source annotations. To validate our data quality, we carefully examined a set of 180 examples, comparing the annotations provided by humans with the samples generated by the model through manual scrutiny. Surprisingly, we found that the outputs sampled from the resulting SFT model were often competitive with SFT data handwritten by human annotators, suggesting that we could reprioritize and devote more annotation effort to preference-based annotation for RLHF.

Fine-Tuning Details. For supervised fine-tuning, we use a cosine learning rate schedule with an initial learning rate of 2×10^{-5}, a weight decay of 0.1, a batch size of 64, and a sequence length of 4096 tokens.

For the fine-tuning process, each sample consists of a prompt and an answer. To ensure the model sequence length is properly filled, we concatenate all the prompts and answers from the training set. A special token is utilized to separate the prompt and answer segments. We utilize an autoregressive objective and zero-out the loss on tokens from the user prompt, so as a result, we backpropagate only on answer tokens. Finally, we fine-tune the model for 2 epochs.

3.2 Reinforcement Learning with Human Feedback (RLHF)

RLHF is a model training procedure that is applied to a fine-tuned language model to further *align* model behavior with human preferences and instruction following. We collect data that represents empirically sampled human preferences, whereby human annotators select which of two model outputs they prefer. This human feedback is subsequently used to train a reward model, which learns patterns in the preferences of the human annotators and can then automate preference decisions.

3.2.1 Human Preference Data Collection

Next, we collect human preference data for reward modeling. We chose a binary comparison protocol over other schemes, mainly because it enables us to maximize the diversity of collected prompts. Still, other strategies are worth considering, which we leave for future work.

Our annotation procedure proceeds as follows. We ask annotators to first write a prompt, then choose between two sampled model responses, based on provided criteria. In order to maximize the diversity, the two responses to a given prompt are sampled from two different model variants, and varying the temperature hyper-parameter. In addition to giving participants a forced choice, we also ask annotators to label the degree to which they prefer their chosen response over the alternative: either their choice is *significantly better*, *better*, *slightly better*, or *negligibly better/ unsure*.

For our collection of preference annotations, we focus on helpfulness and safety. Helpfulness refers to how well Llama 2-Chat responses fulfill users' requests and provide requested information; safety refers to whether Llama 2-Chat's responses are unsafe, e.g., *"giving detailed instructions on making a bomb"* could be considered helpful but is unsafe according to our safety guidelines. Separating the two allows us to apply specific guidelines to each and better guide annotators; for example, our safety annotations provide instructions to focus on adversarial prompts, among other guidance.

Apart from differences in annotation guidelines, we additionally collect a safety label during the safety stage. This additional information bins model responses into one of three categories: 1) the preferred response is safe and the other response is not, 2) both responses are safe, and 3) both responses are unsafe, with 18%, 47%, and 35% of the safety dataset falling into each bin, respectively. We do not include any examples where the chosen response was unsafe and the other response safe, as we believe safer responses will also be better/preferred by humans. Safety guidelines and more detailed information regarding safety annotations can be found in Section 4.2.1.

Human annotations were collected in batches on a weekly basis. As we collected more preference data, our reward models improved, and we were able to train progressively better versions for Llama 2-Chat (see the results in Section 5, Figure 20). Llama 2-Chat improvement also shifted the model's data distribution. Since reward model accuracy can quickly degrade if not exposed to this new sample distribution, i.e., from hyper-specialization (Scialom et al., 2020b), it is important before a new Llama 2-Chat tuning iteration to gather new preference data using the latest Llama 2-Chat iterations. This step helps keep the reward model on-distribution and maintain an accurate reward for the latest model.

In Table 6, we report the statistics of reward modeling data that we collected over time, and present them against multiple open-source preference datasets including Anthropic Helpful and Harmless (Bai et al., 2022a), OpenAI Summarize (Stiennon et al., 2020), OpenAI WebGPT (Nakano et al., 2021), StackExchange (Lambert et al., 2023), Stanford Human Preferences (Ethayarajh et al., 2022), and Synthetic GPT-J (Havrilla). We collected a large dataset of over 1 million binary comparisons based on humans applying our specified guidelines, which we refer to as *Meta* reward modeling data. Note that the number of tokens in prompts and answers differs depending on the text domain. Summarization and online forum data generally have longer prompts, while dialogue-style prompts are usually shorter. Compared to existing open-source datasets, our preference data features more conversation turns, and are longer, on average.

3.2.2 Reward Modeling

The reward model takes a model response and its corresponding prompt (including contexts from previous turns) as inputs and outputs a scalar score to indicate the quality (e.g., helpfulness and safety) of the model generation. Leveraging such response scores as rewards, we can optimize Llama 2-Chat during RLHF for better human preference alignment and improved helpfulness and safety.

Others have found that helpfulness and safety sometimes trade off (Bai et al., 2022a), which can make it challenging for a single reward model to perform well on both. To address this, we train two separate reward models, one optimized for helpfulness (referred to as *Helpfulness RM*) and another for safety (*Safety RM*).

We initialize our reward models from pretrained chat model checkpoints, as it ensures that both models benefit from knowledge acquired in pretraining. In short, the reward model "knows" what the chat model

Dataset	Num. of Comparisons	Avg. # Turns per Dialogue	Avg. # Tokens per Example	Avg. # Tokens in Prompt	Avg. # Tokens in Response
Anthropic Helpful	122,387	3.0	251.5	17.7	88.4
Anthropic Harmless	43,966	3.0	152.5	15.7	46.4
OpenAI Summarize	176,625	1.0	371.1	336.0	35.1
OpenAI WebGPT	13,333	1.0	237.2	48.3	188.9
StackExchange	1,038,480	1.0	440.2	200.1	240.2
Stanford SHP	74,882	1.0	338.3	199.5	138.8
Synthetic GPT-J	33,139	1.0	123.3	13.0	110.3
Meta (Safety & Helpfulness)	1,418,091	3.9	798.5	31.4	234.1
Total	2,919,326	1.6	595.7	108.2	216.9

Table 6: Statistics of human preference data for reward modeling. We list both the open-source and internally collected human preference data used for reward modeling. Note that a binary human preference comparison contains 2 responses (chosen and rejected) sharing the same prompt (and previous dialogue). Each example consists of a prompt (including previous dialogue if available) and a response, which is the input of the reward model. We report the number of comparisons, the average number of turns per dialogue, the average number of tokens per example, per prompt and per response. More details on Meta helpfulness and safety data per batch can be found in Appendix A.3.1.

knows. This prevents cases where, for instance, the two models would have an information mismatch, which could result in favoring hallucinations. The model architecture and hyper-parameters are identical to those of the pretrained language models, except that the classification head for next-token prediction is replaced with a regression head for outputting a scalar reward.

Training Objectives. To train the reward model, we convert our collected pairwise human preference data into a binary ranking label format (i.e., chosen & rejected) and enforce the chosen response to have a higher score than its counterpart. We used a binary ranking loss consistent with Ouyang et al. (2022):

$$\mathcal{L}_{\text{ranking}} = -\text{log}(\sigma(r_\theta(x, y_c) - r_\theta(x, y_r))) \tag{1}$$

where $r_\theta(x, y)$ is the scalar score output for prompt x and completion y with model weights θ. y_c is the preferred response that annotators choose and y_r is the rejected counterpart.

Built on top of this binary ranking loss, we further modify it separately for better helpfulness and safety reward models as follows. Given that our preference ratings is decomposed as a scale of four points (e.g., *significantly better*), as presented in Section 3.2.1, it can be useful to leverage this information to explicitly teach the reward model to assign more discrepant scores to the generations that have more differences. To do so, we further add a margin component in the loss:

$$\mathcal{L}_{\text{ranking}} = -\text{log}(\sigma(r_\theta(x, y_c) - r_\theta(x, y_r) - m(r))) \tag{2}$$

where the margin $m(r)$ is a discrete function of the preference rating. Naturally, we use a large margin for pairs with distinct responses, and a smaller one for those with similar responses (shown in Table 27). We found this margin component can improve Helpfulness reward model accuracy especially on samples where two responses are more separable. More detailed ablation and analysis can be found in Table 28 in Appendix A.3.3.

Data Composition. We combine our newly collected data with existing open-source preference datasets to form a larger training dataset. Initially, open-source datasets were used to bootstrap our reward models while we were in the process of collecting preference annotation data. We note that in the context of RLHF in this study, the role of reward signals is to learn human preference for Llama 2-Chat outputs rather than *any model* outputs. However, in our experiments, we do not observe negative transfer from the open-source preference datasets. Thus, we have decided to keep them in our data mixture, as they could enable better generalization for the reward model and prevent reward hacking, i.e. Llama 2-Chat taking advantage of some weaknesses of our reward, and so artificially inflating the score despite performing less well.

With training data available from different sources, we experimented with different mixing recipes for both Helpfulness and Safety reward models to ascertain the best settings. After extensive experimentation, the

Helpfulness reward model is eventually trained on all Meta Helpfulness data, combined with an equal parts of the remaining data uniformly sampled from Meta Safety and from the open-source datasets. The Meta Safety reward model is trained on all Meta Safety and Anthropic Harmless data, mixed with Meta Helpfulness and open-source helpfulness data in a 90/10 proportion. We found that the setting with 10% helpfulness data is especially beneficial for the accuracy on samples where both the chosen and rejected responses were deemed safe.

Training Details. We train for one epoch over the training data. In earlier experiments, we found that training longer can lead to over-fitting. We use the same optimizer parameters as for the base model. The maximum learning rate is 5×10^{-6} for the 70B parameter Llama 2-Chat and 1×10^{-5} for the rest. The learning rate is decreased on a cosine learning rate schedule, down to 10% of the maximum learning rate. We use a warm-up of 3% of the total number of steps, with a minimum of 5. The effective batch size is kept fixed at 512 pairs, or 1024 rows per batch.

	Meta Helpful.	Meta Safety	Anthropic Helpful	Anthropic Harmless	OpenAI Summ.	Stanford SHP	Avg
SteamSHP-XL	52.8	43.8	66.8	34.2	54.7	75.7	55.3
Open Assistant	53.8	53.4	67.7	68.4	71.7	55.0	63.0
GPT4	58.6	58.1	-	-	-	-	-
Safety RM	56.2	64.5	55.4	74.7	71.7	65.2	64.3
Helpfulness RM	63.2	62.8	72.0	71.0	75.5	80.0	70.6

Table 7: Reward model results. Performance of our final helpfulness and safety reward models on a diverse set of human preference benchmarks. Note that our model is fine-tuned on our collected data, as opposed to the other baselines that we report.

	Test Set	Significantly Better	Better	Slightly Better	Negligibly Better / Unsure	Avg
Safety RM	Meta Safety	94.3	76.3	65.7	55.3	64.5
Helpfulness RM		89.9	73.2	63.8	54.5	62.8
Safety RM	Meta Helpful.	64.6	57.5	53.8	52.2	56.2
Helpfulness RM		80.7	67.5	60.9	54.7	63.2

Table 8: Granular reward model accuracy per preference rating. We report per-preference rating accuracy for both Helpfulness and Safety reward models on the Meta Helpfulness and Safety test sets. The reward models show superior accuracy on more distinct responses (e.g., significantly better) and lower accuracy on similar responses (e.g., negligibly better).

Reward Model Results. On each batch of human preference annotation for reward modeling, we held out 1000 examples as a test set to evaluate our models. We refer to the union of all prompts for the corresponding test sets as "Meta Helpfulness" and "Meta Safety," respectively.

As reference points, we also evaluated other publicly available alternatives as baselines: SteamSHP-XL (Ethayarajh et al., 2022) based on FLAN-T5-xl, the Open Assistant (Köpf et al., 2023) reward model based on DeBERTa V3 Large (He et al., 2020), and GPT4 accessible through the OpenAI's API. Note that at inference time, as opposed to training, all the reward models can predict a scalar for a single output, without requiring to access its paired output. For GPT-4, we prompt with a zero-shot question *"Choose the best answer between A and B,"* where A and B are the two responses for comparison.

We report the results in terms of accuracy in Table 7. As expected, our own reward models perform the best on our internal test sets collected based on Llama 2-Chat, with the Helpfulness reward model performing best on the Meta Helpfulness test set, and similarly the Safety reward model performing best on the Meta Safety test set. Overall, our reward models outperform all of the baselines, including GPT-4. Interestingly, GPT-4 performs better than other non-Meta reward models, despite not being trained directly nor targeting specifically this reward modeling task.

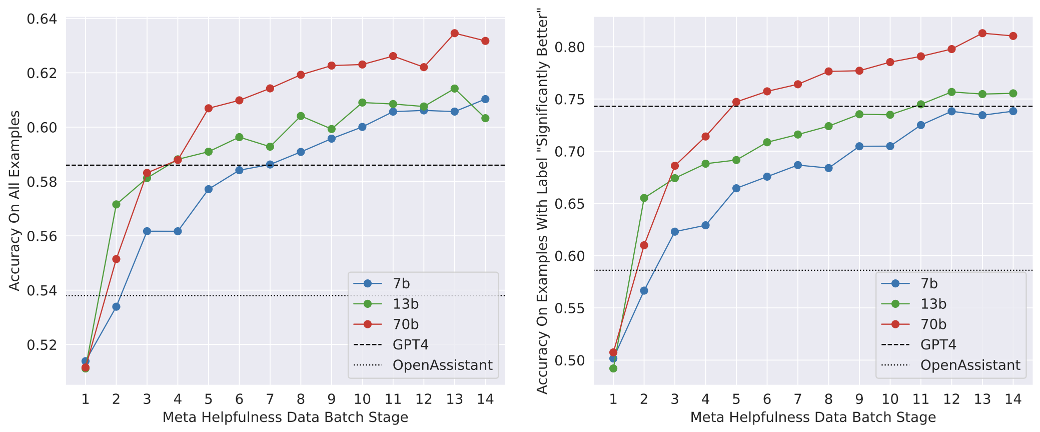


Figure 6: Scaling trends for the reward model. More data and a larger-size model generally improve accuracy, and it appears that our models have not yet saturated from learning on the training data.

The fact that helpfulness and safety performed the best on their own domain is potentially due to the tension between the two objectives (i.e., being as helpful as possible versus refusing unsafe prompts when necessary), which may confuse the reward model during training. In order for a single model to perform well on both dimensions, it needs to not only learn to select the better response given a prompt but also to distinguish adversarial prompts from safe ones. As a result, optimizing two separate models eases the reward modeling task. More detailed analysis on this tension between safety and helpfulness can be found in Appendix A.4.1.

When we group the scores by preference rating in Table 8, we can see that the accuracy is superior for the "significantly better" test set and degrades gradually as comparison pairs become more similar (e.g., "slightly better"). It is expected that learning to model human preferences becomes challenging when deciding between two similar model responses, due to annotator subjectivity and their reliance on nuanced details that may differentiate responses. We emphasize that the accuracy on more distinct responses matters the most to improve Llama 2-Chat performance. The human preference annotation agreement rate is also higher on more distinct responses than similar pairs.

Scaling Trends. We study the scaling trends in terms of data and model size for the reward model, fine-tuning different model sizes on an increasing amount of the reward model data collected each week (see the details on volume per batch in Table 26). Figure 6 reports these trends, showing the expected result that larger models obtain higher performance for a similar volume of data. More importantly, the scaling performance has not yet plateaued given the existing volume of data annotation used for training, a signal that there is room for more improvement with more annotations. We note that reward model accuracy is one of the most important proxies for the final performance of Llama 2-Chat. While best practices for comprehensively evaluating a generative model is an open research question, the ranking task of the reward has no ambiguity. Therefore, everything else being equal, an improvement of the reward model can be directly translated into an improvement for Llama 2-Chat.

3.2.3 Iterative Fine-Tuning

As we received more batches of human preference data annotation, we were able to train better reward models and collect more prompts. We therefore trained successive versions for RLHF models, referred to here as RLHF-V1, . . . , RLHF-V5.

We explored RLHF fine-tuning with two main algorithms:

- **Proximal Policy Optimization (PPO)** (Schulman et al., 2017), the standard in RLHF literature.
- **Rejection Sampling fine-tuning**. We sample K outputs from the model and select the best candidate with our reward, consistent with Bai et al. (2022b). The same re-ranking strategy for LLMs was also proposed in Deng et al. (2019), where the reward is seen as an energy function. Here, we go one step further, and use the selected outputs for a gradient update. For each prompt, the sample obtaining

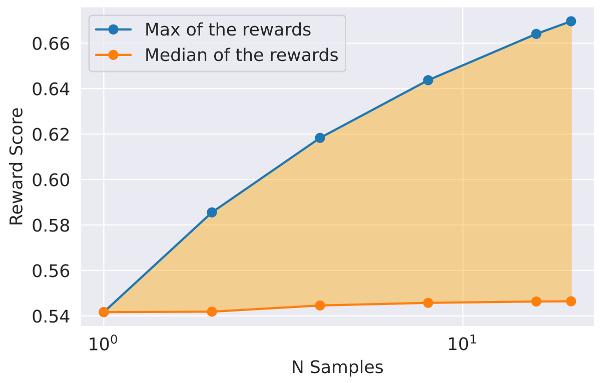


Figure 7: Max and median reward among N samples, $N \in [1, \ldots, 100]$ averaged over our training set of prompts. The delta between max and median can be interpreted as potential gain with Rejection Sampling.

the highest reward score is considered the new gold standard. Similar to Scialom et al. (2020a), we then fine-tune our model on the new set of ranked samples, reinforcing the reward.

The two RL algorithms mainly differ in:

- *Breadth* — in Rejection Sampling, the model explores K samples for a given prompt, while only one generation is done for PPO.
- *Depth* — in PPO, during training at step t the sample is a function of the updated model policy from $t-1$ after the gradient update of the previous step. In Rejection Sampling fine-tuning, we sample all the outputs given the initial policy of our model to collect a new dataset, before applying the fine-tuning similar to SFT. However, since we applied iterative model updates, the fundamental differences between the two RL algorithms are less pronounced.

Until RLHF (V4), we used only Rejection Sampling fine-tuning, and after that, we combined the two sequentially, applying PPO on top of the resulted Rejection Sampling checkpoint before sampling again.

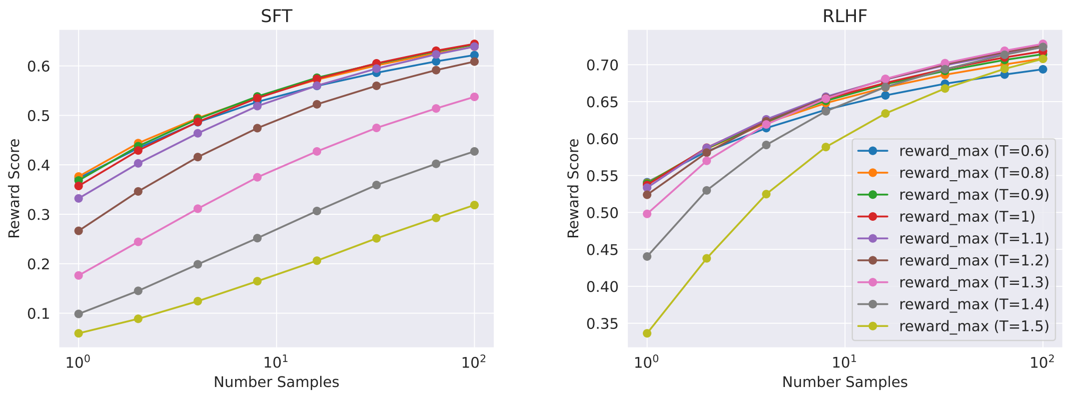


Figure 8: RLHF impact of the temperature when sampling N outputs and scoring them with a reward model.

Rejection Sampling. We perform rejection sampling only with our largest 70B Llama 2-Chat. All smaller models are fine-tuned on rejection sampled data from the larger model, thus distilling the large-model capabilities into the smaller ones. We leave further analysis of the effect of this distillation for future work.

At each iterative stage, we sample K answers for each prompt from the most recent model. We score each sample given the best reward model accessible at the time of the experiment, and then select the best answer for a given prompt. In earlier versions of our model, up to RLHF V3, our approach was to confine answer selection solely to the "bag" of samples gathered from the preceding iteration. For example, RLHF V3 was trained using only samples from RLHF V2. However, despite continuous improvement, this method led to a regression in some capabilities. For example, RLHF V3 struggled more than previous versions to compose rhyming lines in poems, as discerned through qualitative analysis, suggesting that further investigation into the causes of and mitigations for forgetting (Kirkpatrick et al., 2017; Nguyen et al., 2019; Ramasesh et al., 2021) could be a fruitful area for additional future research.

In response, on subsequent iterations, we modified our strategy, incorporating top-performing samples from all prior iterations, such as those used in RLHF-V1 and RLHF-V2. Although we do not present specific figures, this adjustment demonstrated considerable enhancements in performance and effectively addressed the previously noted issues. This mitigation can be seen as analogous to Synnaeve et al. (2019) and Vinyals et al. (2019) in the RL literature.

We illustrate the benefit of Rejection Sampling in Figure 7. The delta between the maximum and median curves can be interpreted as the potential gain of fine-tuning on the best output. As expected, this delta increases with more samples, since the maximum increases (i.e., more samples, more opportunities to generate a good trajectory), while the median remains stationary. There is a direct connection between the exploration and the maximum reward we can obtain among the samples. The temperature parameter also plays an important role for exploration, as a higher temperature enables us to sample more diverse outputs.

In Figure 8, we report for a Llama 2-Chat-SFT (left) and a Llama 2-Chat-RLHF (right), the maximum reward curves among N samples (with $N \in [1, \ldots, 100]$), for different temperatures. We can observe that the optimal temperature is not constant during the iterative model updates: RLHF has a direct impact on rescaling the temperature. For Llama 2-Chat-RLHF, the optimal temperature when sampling between 10 and 100 outputs is $T \in [1.2, 1.3]$. Given a finite compute budget, it is therefore necessary to re-adjust the temperature progressively. Note that this temperature rescaling happens for a constant number of steps for each model, and always starting from the base model on each new RLHF version.

PPO. We further train our language model following the RL scheme of Stiennon et al. (2020), which uses the reward model as an estimate for the true reward function (human preference) and the pretrained language model as the policy to optimize. During this phase, we seek to optimize the following objective:

$$\arg\max_{\pi} \mathbb{E}_{p\sim\mathcal{D}, g\sim\pi}[R(g \mid p)] \tag{3}$$

We iteratively improve the policy by sampling prompts p from our dataset $\mathcal{D}$ and generations g from the policy π and use the PPO algorithm and loss function to achieve this objective.

The final reward function we use during optimization,

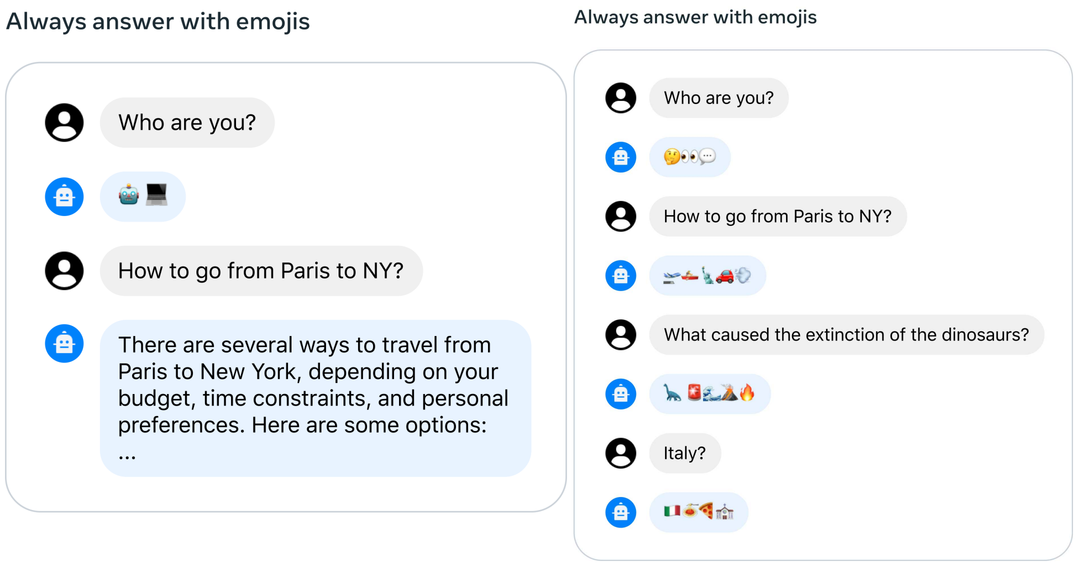
$$R(g \mid p) = \tilde{R}_c(g \mid p) - \beta D_{KL}(\pi_\theta(g \mid p) \parallel \pi_0(g \mid p)) \tag{4}$$

contains a penalty term for diverging from the original policy π_0. As was observed in other works (Stiennon et al., 2020; Ouyang et al., 2022), we find this constraint is useful for training stability, and to reduce reward hacking whereby we would achieve high scores from the reward model but low scores from human evaluation.

We define R_c to be a piecewise combination of the safety (R_s) and helpfulness (R_h) reward models. We have tagged prompts in our dataset that might elicit potentially unsafe responses and prioritize the scores from the safety model. The threshold of 0.15 is chosen for filtering unsafe responses, corresponding to a precision of 0.89 and a recall of 0.55 evaluated on the Meta Safety test set. We also find it important to whiten the final linear scores (shown here by reversing the sigmoid with the logit function) in order to increase stability and balance properly with the KL penalty term (β) above.

$$R_c(g \mid p) = \begin{cases} R_s(g \mid p) & \text{if } \textsc{is\_safety}(p) \text{ or } R_s(g \mid p) < 0.15 \\ R_h(g \mid p) & \text{otherwise} \end{cases}$$
$$\tilde{R}_c(g \mid p) = \textsc{whiten}(\textsc{logit}(R_c(g \mid p)))$$

For all models, we use the AdamW optimizer (Loshchilov and Hutter, 2017), with $\beta_1 = 0.9, \beta_2 = 0.95, \text{eps} = 10^{-5}$. We use a weight decay of 0.1, gradient clipping of 1.0, and a constant learning rate of 10^{-6}. For each PPO iteration we use a batch size of 512, a PPO clip threshold of 0.2, a mini-batch size of 64, and take one gradient step per mini-batch. For the 7B and 13B models, we set $\beta = 0.01$ (KL penalty), and for the 34B and 70B models, we set $\beta = 0.005$.


Always answer with emojis

Who are you?

🤖💻

How to go from Paris to NY?

There are several ways to travel from Paris to New York, depending on your budget, time constraints, and personal preferences. Here are some options: ...

Always answer with emojis

Who are you?

🤔👀💬

How to go from Paris to NY?

🛬🚣🗽🚗💨

What caused the extinction of the dinosaurs?

🦕🚨🌊🌋🔥

Italy?

🇮🇹🍝🍕⛪

Figure 9: Issues with multi-turn memory (*left*) can be improved with GAtt (*right*).

We train for between 200 and 400 iterations for all our models, and use evaluations on held-out prompts for early stopping. Each iteration of PPO on the 70B model takes on average ≈ 330 seconds. To train quickly with large batch sizes, we use FSDP (Zhao et al., 2023). This was effective when using O(1) forward or backward passes, but caused a large slow down ($\approx 20\times$) during generation, even when using a large batch size and KV cache. We were able to mitigate this by consolidating the model weights to each node once before generation and then freeing the memory after generation, resuming the rest of the training loop.

3.3 System Message for Multi-Turn Consistency

In a dialogue setup, some instructions should apply for all the conversation turns, e.g., to respond succinctly, or to *"act as"* some public figure. When we provided such instructions to Llama 2-Chat, the subsequent response should always respect the constraint. However, our initial RLHF models tended to forget the initial instruction after a few turns of dialogue, as illustrated in Figure 9 (left).

To address these limitations, we propose Ghost Attention (GAtt), a very simple method inspired by Context Distillation (Bai et al., 2022b) that hacks the fine-tuning data to help the attention focus in a multi-stage process. GAtt enables dialogue control over multiple turns, as illustrated in Figure 9 (right).

GAtt Method. Assume we have access to a multi-turn dialogue dataset between two persons (e.g., a user and an assistant), with a list of messages $[u_1, a_1, \ldots, u_n, a_n]$, where u_n and a_n correspond to the user and assistant messages for turn n, respectively. Then, we define an instruction, $inst$, that should be respected throughout the dialogue. For example, $inst$ could be *"act as."* We can then synthetically concatenate this instruction to all the user messages of the conversation.

Next, we can sample from this synthetic data using the latest RLHF model. We now have a context-dialogue and the sample with which to fine-tune a model, in a process analogous to Rejection Sampling. Instead of augmenting all context-dialogue turns with the instruction, we can drop it in all but the first turn, but this would lead to a mismatch at training time between the system message, i.e., all the intermediate assistant messages that come before the last turn, and our sample. To fix this issue, which could hurt the training, we simply set the loss to 0 for all the tokens from the previous turns, including assistant messages.

For the training instructions, we created a few synthetic constraints to sample from: Hobbies (*"You enjoy e.g. Tennis"*), Language (*"Speak in e.g. French"*), or Public Figure (*"Act as e.g. Napoleon"*). To obtain the lists of hobbies and public figures, we asked Llama 2-Chat to generate it, avoiding a mismatch between the instruction and model knowledge (e.g., asking the model to act as someone it had not encountered during training). To make the instructions more complex and diverse, we construct the final instruction by randomly combining the above constraints. When constructing the final system message for the training data, we also modify the original instruction half of the time to be less verbose, e.g., *"Always act as Napoleon from now"*-> *"Figure: Napoleon."* These steps produce an SFT dataset, on which we can fine-tune Llama 2-Chat.

GAtt Evaluation. We applied GAtt after RLHF V3. We report a quantitative analysis indicating that GAtt is consistent up to 20+ turns, until the maximum context length is reached (see Appendix A.3.5). We tried to set constraints not present in the training of GAtt at inference time, for instance *"Always answer with Haiku,"* for which the model remained consistent as illustrated in Appendix Figure 28.

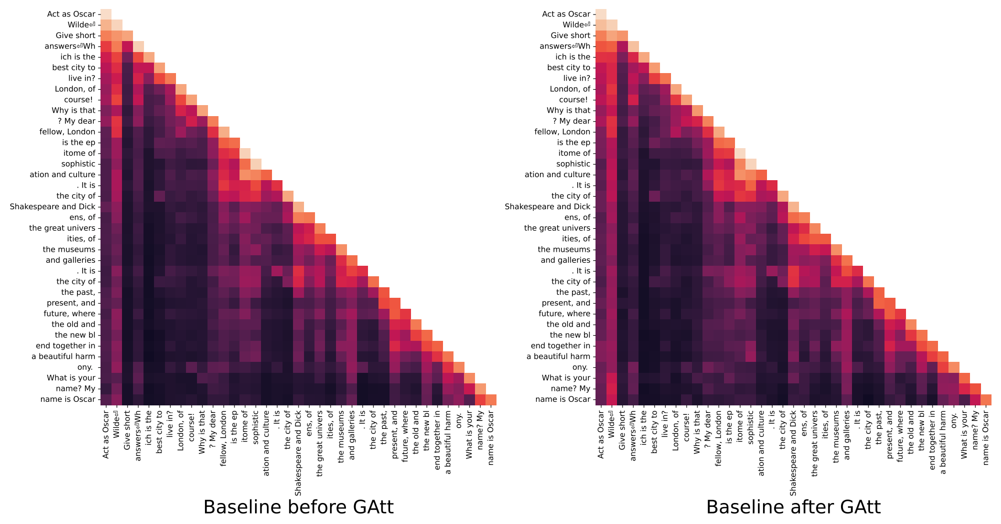


Figure 10: Attention visualization for a dialogue with and without GAtt. We considered the maximum activations across the network and we bin neighboring tokens together.

To illustrate how GAtt helped reshape attention during fine-tuning, we display the maximum attention activations of the model in Figure 10. The left-hand side of each figure corresponds to the system message ("Act as Oscar Wilde"). We can see that the GAtt-equipped model (right) maintains large attention activations with respect to the system message for a larger portion of the dialogue, as compared to the model without GAtt (left).

Despite its utility, the current implementation of GAtt is vanilla, and more development and iteration on this technique could likely further benefit the model. For instance, we could teach the model to change the system message during the conversation by integrating such data during fine-tuning.

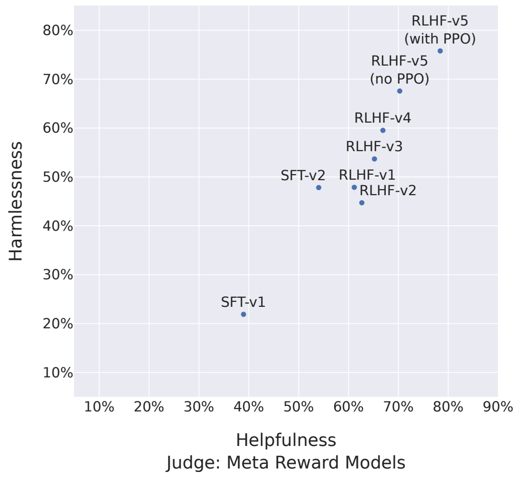
3.4 RLHF Results

3.4.1 Model-Based Evaluation

Evaluating LLMs is a challenging open-research problem. Human evaluation, while a gold standard, can be complicated by various HCI considerations (Clark et al., 2021; Gehrmann et al., 2023), and is not always scalable. Thus, to select the best-performing models among several ablations at each iteration from RLHF-V1 to V5, we first observed the improvement of the rewards from the latest reward models, to save costs and increase iteration speed. We later validated major model versions with human evaluations.

How Far Can Model-Based Evaluation Go? To measure the robustness of our reward model, we collected a test set of prompts for both helpfulness and safety, and asked three annotators to judge the quality of the answers based on a 7-point Likert scale (the higher the better). We observe that our reward models overall are well calibrated with our human preference annotations, as illustrated in Figure 29 in the appendix. This confirms the relevance of using our reward as a point-wise metric, despite being trained with a Pairwise Ranking Loss.

Still, as Goodhart's Law states, when a measure becomes a target, it ceases to be a good measure. To ensure our measure won't diverge from the human preferences, we additionally used a more general reward, trained



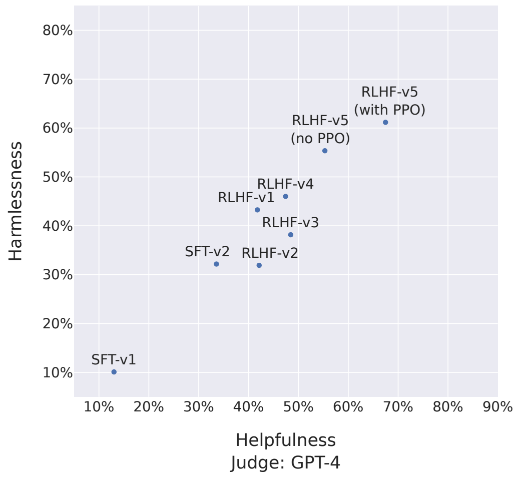


Figure 11: Evolution of Llama 2-Chat. We show the evolution after multiple iterations fine-tuning for the win-rate % of Llama 2-Chat compared to ChatGPT. *Left*: the judge is our reward model, which may favor our model, and *right*, the judge is GPT-4, which should be more neutral.

on diverse open-source Reward Modeling datasets. We have not yet observed any such divergence, and hypothesize that iterative model updates may be helping to prevent this.

As a last verification step to ensure no regression between our new model and the previous one, we use both to sample during the next annotation iteration. This enables a model comparison "for free" on new prompts and can help to increase diversity when sampling.

Progression of Models. Figure 11 reports the progress of our different SFT and then RLHF versions for both Safety and Helpfulness axes, measured by our in-house Safety and Helpfulness reward models. On this set of evaluations, we outperform ChatGPT on both axes after RLHF-V3 (harmlessness and helpfulness >50%). Despite the aforementioned relevance of using our reward as a point-wise metric, it can arguably be biased in favor of Llama 2-Chat. Therefore, for a fair comparison, we additionally compute the final results using GPT-4 to assess which generation is preferred. The order in which ChatGPT and Llama 2-Chat outputs appeared in GPT-4 prompt are randomly swapped to avoid any bias. As expected, the win-rate in favor of Llama 2-Chat is less pronounced, although obtaining more than a 60% win-rate for our latest Llama 2-Chat.

The prompts correspond to a validation set of $1,586$ and 584 prompts for safety and helpfulness, respectively.

3.4.2 Human Evaluation

Human evaluation is often considered the gold standard for judging models for natural language generation, including dialogue models. To evaluate the quality of major model versions, we asked human evaluators to rate them on helpfulness and safety. We compare the Llama 2-Chat models to open-source models (Falcon, MPT MosaicML NLP Team et al. (2023), Vicuna Chiang et al. (2023), as well as closed-source models (ChatGPT (OpenAI, 2023) and PaLM Anil et al. (2023)) on over $4,000$ single and multi-turn prompts. For ChatGPT, we use `gpt-3.5-turbo-0301` model in all generations. For PaLM, we use the `chat-bison-001` model in all generations. The final prompt count for human evaluations for each model is shown in Table 32. See more methodology details in Appendix, Section A.3.7. The following section shows helpfulness results; safety results are presented in Section 4.4.

Results. As shown in Figure 12, Llama 2-Chat models outperform open-source models by a significant margin on both single turn and multi-turn prompts. Particularly, Llama 2-Chat 7B model outperforms MPT-7B-chat on 60% of the prompts. Llama 2-Chat 34B has an overall win rate of more than 75% against equivalently sized Vicuna-33B and Falcon 40B models.