AUTOALIGN: Get Your LLM Aligned with Minimal Annotations

https://github.com/icip-cas/AutoAlign

Xinyu Lu*, Dong Xu*, Chunkang Zhang*,
Xinyan Guan, Junxiang Wang, Qingyu Zhang, Pengbo Wang,
Yingzhi Mao, Hao Xiang, Xueru Wen, Zichao Li,
Yaojie Lu†, Hongyu Lin†, Le Sun, Xianpei Han
[1]Chinese Information Processing Laboratory,
Institute of Software, Chinese Academy of Sciences
[2]University of Chinese Academy of Sciences
{luxinyu2021,luyaojie,hongyu,sunle,xianpei}@iscas.ac.cn

Abstract

Automated Alignment (Cao et al., 2024) refers to a set of algorithms designed to align Large Language Models (LLMs) with human intentions and values while minimizing manual intervention. However, it faces challenges such as algorithmic diversity and excessively convoluted workflows. We present AUTOALIGN, an open-source toolkit that offers: (1) a unified framework integrating mainstream automated algorithms through a consistent interface, and (2) an accessible workflow supporting one-click execution for prompt synthesis, automatic alignment signal construction, and iterative model training. Our toolkit enables easy reproduction of existing results through extensive benchmarks and facilitates the development of novel approaches via modular components. It includes implementations for both highly efficient inference and training, as well as low-resource training. By standardizing automated alignment methodologies and providing accessible implementations, AUTOALIGN lowers the barriers to building customized aligned models and supports academic research.

	AUTO ALIGN	Llama Factory	Easy Instruct	Auto Train	Prompt 2Model
Data Syn.	✓		✓		✓
Data Manage.	✓	✓	✓		✓
Training	✓	✓		✓	✓
Evaluation	✓	✓			✓
Deployment	✓	✓			✓
Megatron	✓				
Pipeline	✓				
Min Anno.	✓		✓		✓

Table 1: Feature Comparison Between AUTOALIGN and Related Frameworks. Syn. denotes synthesis. Anno. denotes annotation.

1 Introduction

In recent years, the development of Large Language Models (LLMs) has advanced rapidly. A key technology that enables these models to be applied in real-world scenarios is alignment, ensuring that the model outputs meet human requirements and adhere to human intentions and values. Alignment techniques, such as Supervised Fine-tuning (SFT) (Wei et al., 2022), Reinforcement Learning from Human Feedback (RLHF) (Ouyang et al., 2022), and Direct Preference Optimization (DPO) (Rafailov et al., 2023), typically involve training models with demonstration or preference data. However, constructing demonstration and preference data requires significant manual labor, leading to scalability challenges and high costs. Consequently, developing alignment algorithms that require less human intervention has received increased attention (Cao et al., 2024).

Meanwhile, as shown in Table 1, researchers have developed a fragmented ecosystem of specialized packages, each operating in isolation to address specific aspects of LLM development such as fine-tuning LLMs, instruction synthesis, and management. For instance, LlamaFactory (Zheng et al., 2024) focuses on supporting diverse fine-tuning methods and efficient tuning for large language models. EasyInstruct (Ou et al., 2024) specializes in prompt synthesis and filtering. AutoTrain (Thakur, 2024) standardizes input-output formats for different training tasks, providing a code-free, unified training interface. The Prompt2Model framework (Viswanathan et al., 2023) enables models to retrieve required data based on user specifications.

However, limited effort has been devoted to unifying these steps and transforming existing methods into simple automated alignment pipelines that developers and researchers can use end-to-end for training and developing models. To this end,

*Equal Contribution.
†Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations), pages 189–198

we develop AUTOALIGN, which provides a unified framework for different automated alignment pipelines, automated evaluation modules, and deployment. We apply necessary abstractions to the various methods, enabling code and function reuse across different pipelines. Additionally, we develop a user interface that allows easy configuration of pipeline, evaluation, and deployment processes in a low-code manner. This intuitive interface and minimal annotation requirements significantly reduce the barrier to entry for researchers and practitioners without specialized machine learning expertise or extensive annotation resources. Furthermore, the unified pipeline architecture accelerates development cycles by eliminating redundant implementation work across different alignment techniques and facilitates easy implementation of new ones.

Overall, AUTOALIGN is built with Python and PyTorch (Paszke et al., 2019). To support such an all-in-one and end-to-end process, AUTOALIGN is built upon and benefits from several high-quality open-source libraries. To efficiently conduct large-scale training and sampling, we select Megatron (Shoeybi et al., 2019) and DeepSpeed (Rajbhandari et al., 2019) as the training backends, with vLLM (Kwon et al., 2023) as the inference engine. We integrated OpenCompass (Contributors, 2023) into our evaluation process, and the basic training classes are adapted from Hugging Face Transformers and TRL (von Werra et al., 2020). UI components are implemented using Streamlit[1].

To validate the effectiveness of AUTOALIGN, we provide examples of practical applications. First, we reproduce several classical alignment algorithms, including RLCD (Yang et al., 2023), CAI (Bai et al., 2022), and Self-Rewarding (Yuan et al., 2024)—three algorithms that enable users to align their base models with minimal annotation requirements. These reproductions provide the academic community with a series of empirical practices and baselines. We also validate the effectiveness of basic training functions on classic alignment datasets (Ding et al., 2023; Cui et al., 2023).

2 AUTOALIGN Framework

The core steps for developing LLMs with minimal annotations can be divided into stages including: *instruction synthesis*, *policy improvement*, *iterative training*, *evaluation*, and *deployment*. In this section, we first introduce the key features of AUTOALIGN in supporting these steps, then describe how users can align a LLM from scratch using a unified interface through a low-code manner.

2.1 Instruction Synthesis

Instructions define the capabilities targeted by users in the alignment process. AUTOALIGN integrates three instruction synthesis methods, enabling users to obtain a large number of instructions with minimal human effort. Users can choose to provide seed datasets, an existing instruction-tuned model, or unsupervised data to generate abundant instructions.

Self-Instruct (Wang et al., 2023) leverages a seed data pool and an instruction-following LLM to generate new instructions, followed by automated quality filtering and deduplication. We support various quality filters to eliminate instructions in specific languages, instructions starting with punctuation, etc., and an n-gram-based similarity filter to remove near-duplicate prompts. We implemented the similarity filter based on `torchmetrics` for higher speed.

Back-Translation (Li et al., 2024) uses an instruction-following model to generate instructions from unlabeled pretraining data. This method is particularly useful when the user wants to harvest domain-specific instructions based on domain corpus.

MAGPIE (Xu et al., 2025) directly samples user instructions by hacking the model template. For instance, we can force the instruction following model to generate responses based on input prompt like `<|start_header_id|>user<|end_header_id|>\n\n`. A key feature of this method is that the user is only required to provide an existing aligned model for this method, without inputting any data.

2.2 Policy Improvement

Policy improvement involves scaling test-time inference based on the current policy to sample higher quality outputs. For example, users can apply the initial policy, reward model, and Best-of-N (BoN) strategy to obtain better responses, or using Context Distillation (Snell et al., 2022) by prepending steering prefixes to LLMs to elicit better or worse responses, creating contrastive signals among others. Efficient large-batch sampling is the core of policy improvement. AUTOALIGN's modular design makes it effortless to call inference

[1]Project repository: https://github.com/streamlit/streamlit

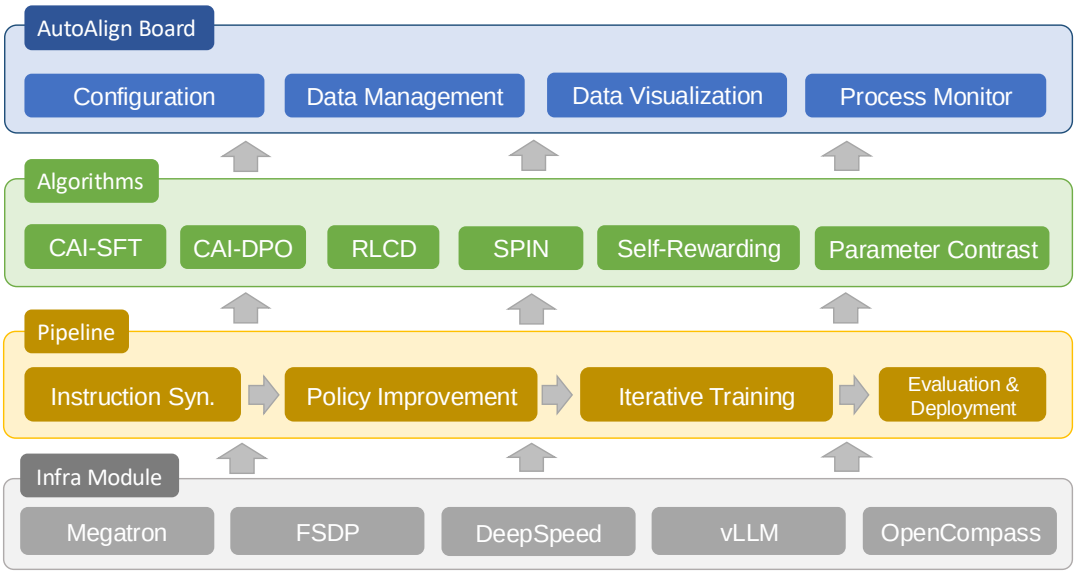

Figure 1: The overview of AUTOALIGN Framework.

engines in any pipeline. In AUTOALIGN, we implement inference based on both HuggingFace and vLLM backend.

Multinode Parallel Inference To accelerate the inference process, we implemented multi-node inference using vLLM based on Ray for multi-node parallelism inferencing. By organizing the GPUs using Ray, we achieved Data Parallelism (DP) on top of vLLM.

2.3 Efficient Iterative Training

The iterative training step in the AUTOALIGN pipeline is to steer the model towards selected sampled responses through the learning process. Besides providing most commonly used full-parameter SFT and DPO training, we also support several training methods, including Megatron-based training, Packing acceleration and PEFT methods.

Megatron Megatron is a PyTorch-based framework designed to overcome the computational challenges of training LLMs with billions of parameters. Unlike traditional data parallelism, Megatron-Core offers comprehensive support for advanced parallelism strategies including tensor, sequence, pipeline, context, and Mixture of Experts (MoE) expert parallelism.

AUTOALIGN provides Megatron-based SFT and DPO implementations. Compared to the popular Hugging Face Trainer, our implementation achieves an approximately four-fold acceleration (Table 2) in training speed for DPO on 70B parameter model, while maintaining comparable model performance. This enhancement demonstrates the practical benefits of integrating Megatron-Core's capabilities into the automated alignment process.

Metric	Megatron	HuggingFace
Training config	72B DPO	
Hardware config	8 nodes, 64 GPUs, 40GB per GPU	
Processing speed	517 it/s	129 it/s
Time to process 5,000 samples	10 min	39 min

Table 2: Performance comparison between Megatron and HuggingFace implementations for 72B DPO model training.

Packing We implement sequence packing (Bai et al., 2024) in AUTOALIGN, concatenating multiple short sequences to maximize length utilization, reducing padding and training steps. By extending Flash Attention 2 (Dao, 2024; Kundu et al., 2024) with integer-based sequence numbering masks rather than binary masks, we prevent attention cross-contamination (Krell et al., 2022) while maintaining efficiency. Experiments show a 56% training speedup without performance loss.

Parameter Efficient Tuning LoRA (Hu et al., 2022) is a parameter efficient fine-tuning technique whose core principle involves approximating parameter updates using the product of two low-rank matrices. The AUTOALIGN package integrates LoRA through the PEFT library (Mangrulkar et al., 2022) by setting hyperparameters in `LoraConfig` and generating adapted models via `get_peft_model()`.

2.4 Evalulation and Deployment

AUTOALIGN integrates an easy configurable automatic evaluation system, allowing users to configure an evaluation task (as shown in Figure 2) with a few parameters to conduct evaluations across 13 representative benchmarks covering four aspects: instruction following, mathematics, coding, and knowledge. Furthermore, AUTOALIGN supports model deployment through a Web UI and command-line interaction program, allowing developers to intuitively experience the model's capabilities.

```yaml
# Name of the model to evaluate
model_name: qwen2.5-7b-ins
# The chat template used in evaluation
template_name: chatml
# The path of the model to evaluate
model_path: Qwen/Qwen2.5-7B-Instruct
# The type of eval data combination
eval_type: subjective
# GPUs occupied by a single model worker
per_model_gpu: 1
# The batch size of a single worker
batch_size: 8
# The inference backend
backend: vllm
```

Figure 2: Example configuration for automatic model evaluation.

2.5 AutoAlign-Board

AUTOALIGN-BOARD is a user interface based on Streamlit that allows users to customize the autoalign process of LLMs without writing any code. It provides an unified Browser-based interface including instruction synthesis, policy improvement, iterative training and evaluation, assisting users to develop aligned LLMs almost from scratch.

Streamlined Configuration The interface offers multiple configuration options for data synthesis, inference, training, and evaluation with sensible defaults for most parameters, simplifying the alignment process while maintaining flexibility.

Data Management and Visualization Users can monitor data quality through visualizations of token distributions, sources, and domains. The interface supports previewing generated instances and filtering synthesized data by various criteria, allowing for customized dataset creation tailored to specific requirements.

Real-time Alignment Monitoring All alignment processes feature live progress tracking. Instruction synthesis and inference logs are displayed directly in the interface, while training shows real-time loss and gradient curves. Evaluation results appear as they become available, providing immediate insights into model performance.

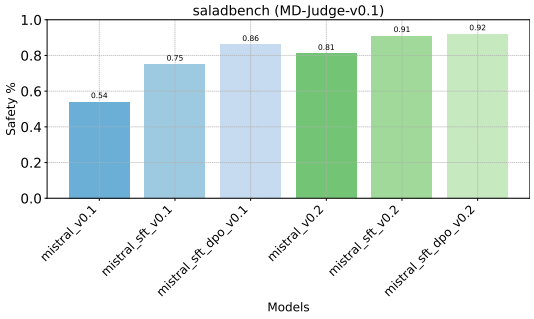

Figure 3: Safety evaluation scores in SaladBench before and after CAI pipeline.

Automated Workflow Navigation The interface intelligently navigates between different stages based on alignment progress, eliminating manual intervention. When processes complete, the interface automatically switches to the appropriate view, ensuring users can monitor the most relevant metrics without manually toggling between pages.

3 Use Practice: Reproduction of Automated Alignment Algorithms

In this section, we demonstrate several attempts to rapidly reproduce representative baseline methods from the alignment research community. These attempts showcase the use of the AUTOALIGN toolkit for weakly supervised alignment. While validating the effectiveness of the toolkit, we believe these attempts will provide valuable references for reproduction by the community.

3.1 Constitutional AI

Constitutional AI (CAI) (Bai et al., 2022) is an approach to train safe, transparent LLMs by defining a set of explicit "constitutions" to guide their behavior. This methodology reduces reliance on manually labeled data and addresses the limitations of traditional supervisory methods. CAI establishes a constitution by defining a set of principles (e.g., "Choose the most helpful, honest, and harmless response.") to guide the model's behavior. The model then self-evaluates and refines its responses based on these principles, instead of relying solely on human feedback, as in traditional RLHF (Askell et al., 2021). The synthesized correction data is used to train the model, enhancing its safety. This

self-improvement requires only a set of queries designed to elicit harmful outputs from the model and a predefined set of human-crafted principles.

In the AUTOALIGN repository, we slightly adjusted some of the CAI settings to better match the capabilities of current language models, while preserving its original concept.

CAI-SFT begins by generating an initial harmful response to queries designed to elicit problematic content. The model then self-critiques its output based on predefined constitutional principles and revises its response accordingly. To ensure consistent generation patterns, we employ few-shot examples at each step. During harmful response generation, we set temperature to 0.7 without the template to encourage exploration, while using templates with temperature 0 for the critique and revision steps. We filter revision responses using quality heuristics—removing those that are too short (<10 characters), too long (>3000 characters), identical to the harmful response, or containing template-specific terms from few-shot examples. The filtered query-revision pairs are then combined with helpful data at a 1:2.5 ratio for supervised fine-tuning to produce the SFT model.

CAI-DPO Building upon CAI-SFT, for preference optimization, we use the SFT model to generate two alternative responses for each query—one with temperature 0 and another with temperature 1. The model is then prompted to evaluate which response better adheres to safety principles. To mitigate position bias, the system swaps response order and performs dual evaluations, with each judgment awarding one point to the response deemed safer. The response accumulating more points is designated as "chosen," while the other becomes "rejected." In cases of tied scores, the system discards the data point to ensure clear preference signals. During the evaluation phase, the system employs few-shot examples and consistent templates to guide the judgment process. The resulting triplets of <query, chosen, rejected> are used to fine-tune the model through DPO, reinforcing constitutional principles through preferences.

As shown in Figure 3, applying the CAI approach boosts the Mistral v0.1 model's safety performance on SALAD-Bench from 0.54 to 0.86, an improvement of over 30 percentage points. The newer Mistral v0.2 model sees a more modest increase, from 0.81 to 0.92, a gain of about 11 percentage points. These results demonstrate how AUTOALIGN significantly improves a model's ability to avoid harmful outputs with only annotated guidelines and self-steering.

3.2 RLCD$_{sys}$

Model	MT-Bench	Ability Sources
Base	5.03	Instructions in Annealing
Instruct	8.15	Complex Post-training Process
UltraChat	7.34	Teacher Distillation
RLCD$_{sys}$	7.29	Self-Steering with System Prompt

Table 3: The RLCD$_{sys}$ variant implemented in AUTOALIGN shows promising results with minimum supervision. All experiments are conducted on the Qwen-2-7B series model.

In the reproduction of RLCD (Reinforcement Learning from Contrastive Distillation) (Yang et al., 2023) algorithm in AUTOALIGN. We use the system prompt region in instruction following model and denote this variant as the RLCD$_{sys}$. In contrast to the standard RLCD approach that relies on explicit "harmful" and "harmless" assistant designations, this variant employs system messages to create diverse contrastive response pairs, offering a more generalizable approach to in-context model steering. The method generates contrastive pairs by presenting the same instruction with two different system messages—one positive and one negative—controlling dimensions such as helpfulness and harmfulness.

The implementation process involves three key stages: data preparation, model steering, and learning. During the steering process, a base instruction-following model (e.g., Qwen2-7B-Base) generates responses conditioned on both positive and negative system messages. For example, a positive system message can be "You should generate an intuitive, user-friendly response," and a negative one can be "You should generate a confusing, user-unfriendly response." To ensure data quality, identical responses between the positive and negative conditions are filtered out to prevent model collapse. The resulting contrastive pairs are then used to train the model using DPO, effectively teaching the model to align with positive system instructions while avoiding behaviors encouraged by negative ones.

Our experimental results (Table 3) demonstrate that RLCD$_{sys}$ achieves general performance comparable to models trained on UltraChat (Ding et al., 2023) data, as measured by the MT-Bench bench-

Model	MT-Bench	IF-Eval (Pr.L)	ARC-e	ARC-c	Hellaswag	GSM8K	MMLU	OpenBookQA	NQ	Exact Acc (%)
Base (M0)	1.86	26.43	69.84	45.42	74.68	55.95	66.62	50.60	16.09	-
IFT (SFT-baseline)	5.46	41.59	74.43	47.46	76.99	57.24	66.36	52.60	29.58	5.08
EFT (M1)	5.48	40.85	70.90	47.80	75.40	57.77	66.27	52.00	29.94	28.44
Self-Rewarding-iter1 (M2)	5.54	41.77	70.90	47.80	75.41	57.62	66.22	52.20	29.86	29.19
Self-Rewarding-iter2 (M3)	5.58	41.96	71.08	48.14	75.41	57.62	66.27	52.20	29.81	27.87

Table 4: Performance of Self-Rewarding models on instruction following, knowledge, reasoning and reward modeling benchmarks. Following the setting of Yuan et al. (2024), all the experiments are conducted with the Llama-3 family of models.

mark (7.29 vs. 7.34). Therefore, RLCD$_{sys}$ showcases the great potential that LLMs can be self-aligned without any demonstration annotation.

3.3 Self-Rewarding

Self-Rewarding Language Model (Yuan et al., 2024) enables a model to autonomously refine its instruction-following capability by using itself as a reward function. This approach generates preference data through self-judgments, reducing reliance on human annotations while unifying reward and generation models for joint optimization through reinforcement learning.

Initialization Our reproduction is based on the LLaMA-3-8B model. We use 3200 examples of high-quality English dialogues from OASST1 (Köpf et al., 2023) for instruction fine-tuning (IFT) to equip the base model with basic instruction-following capability. Additional 1500 examples dialogues with human ratings are used for evaluation fine-tuning (EFT), with a lower learning rate to prevent overfitting as Table 7 shows.

Preference Data Generation We apply self-instruct (Section 2.1) to synthesize prompts. For each prompt, 4 responses are sampled, and the model evaluates each response 3 times, assigning the average score as the reward, following the LLM-as-a-Judge paradigm.

Model Optimization Then we optimize the model with DPO over two iterations, using a decreasing learning rate as Table 7 shows. We found that high weight decay during DPO is crucial for maintaining instruction-following improvements, which may help prevent overfitting when training with limited data (Zhou et al., 2023). Thus, it is set to 0.1 for all Self-Rewarding training stages, including IFT, EFT, and both DPO iterations.

Experimental Results Table 4 presents the performance across different self-rewarding iterations (M0 $\rightarrow$ M1 $\rightarrow$ M2 $\rightarrow$ M3). The results demonstrate incremental gains in instruction-following metrics (MT-Bench and IF-Eval) while maintaining performance on general knowledge and reasoning tasks, confirming the effectiveness of the self-rewarding approach. In addition, we show results on the reserved reward modeling evaluation dataset. Correlation coefficients improved after self-rewarding, indicating that the model aligns more closely with human judgment during the process.

4 Conclusion and Future Work

We presented AUTOALIGN, an open-source toolkit unifying diverse automated alignment techniques under a consistent framework with minimal annotation requirements. Through successful reproductions of RLCD, CAI, and Self-Rewarding, we demonstrated the toolkit's effectiveness in implementing advanced methods with reduced human intervention. AUTOALIGN's standardized abstractions and modular design simplify implementation while facilitating development of novel algorithms, and optimizations like Megatron-based training address key computational challenges. Future work will focus on incorporating emerging automated alignment techniques (Xiang et al., 2024), enhancing multilingual support, and expanding evaluation benchmarks to accelerate progress toward safer, more helpful language models that better serve human needs.

Limitations

Although this demonstration showcases the potential for automated alignment using minimal samples, it still requires human-in-the-loop supervision during the alignment process (e.g., monitoring learning rates, validating output examples, etc.). In future work, we plan to develop an agent system for training models autonomously. Additionally, given the inherent lack of interpretability in training-based alignment methods, we will explore

the use of interpretability techniques for model steering in subsequent development.

Acknowledgments

Many thanks to Boxi Cao for his insightful suggestions and Kaiqi Zhang for his recent contributions to the code repo. We sincerely thank the reviewers for their insightful comments and valuable suggestions. This work was supported by Beijing Natural Science Foundation (L243006), Beijing Municipal Science and Technology Project (Nos. Z231100010323002), the Natural Science Foundation of China (No. 62306303, 62476265). The authors would like to thank Huawei Ascend Cloud Ecological Development Project for the support of Ascend 910 processors.

References

Amanda Askell, Yuntao Bai, Anna Chen, Dawn Drain, Deep Ganguli, Tom Henighan, Andy Jones, Nicholas Joseph, Ben Mann, Nova DasSarma, et al. 2021. A general language assistant as a laboratory for alignment. *arXiv preprint arXiv:2112.00861*.

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

Yushi Bai, Xin Lv, Jiajie Zhang, Yuze He, Ji Qi, Lei Hou, Jie Tang, Yuxiao Dong, and Juanzi Li. 2024. Longalign: A recipe for long context alignment of large language models. *Preprint*, arXiv:2401.18058.

Boxi Cao, Keming Lu, Xinyu Lu, Jiawei Chen, Mengjie Ren, Hao Xiang, Peilin Liu, Yaojie Lu, Ben He, Xianpei Han, et al. 2024. Towards scalable automated alignment of llms: A survey. *arXiv preprint arXiv:2406.01252*.

OpenCompass Contributors. 2023. Opencompass: A universal evaluation platform for foundation models. https://github.com/open-compass/opencompass.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Bingxiang He, Wei Zhu, Yuan Ni, Guotong Xie, Ruobing Xie, Yankai Lin, et al. 2023. Ultrafeedback: Boosting language models with scaled ai feedback. *arXiv preprint arXiv:2310.01377*.

Tri Dao. 2024. FlashAttention-2: Faster attention with better parallelism and work partitioning. In *International Conference on Learning Representations (ICLR)*.

Ning Ding, Yulin Chen, Bokai Xu, Yujia Qin, Zhi Zheng, Shengding Hu, Zhiyuan Liu, Maosong Sun, and Bowen Zhou. 2023. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*.

Edward J Hu, yelong shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2022. LoRA: Low-rank adaptation of large language models. In *International Conference on Learning Representations*.

Mario Michael Krell, Matej Kosec, Sergio P. Perez, and Andrew Fitzgibbon. 2022. Efficient sequence packing without cross-contamination: Accelerating large language models without impacting performance. *Preprint*, arXiv:2107.02027.

Achintya Kundu, Rhui Dih Lee, Laura Wynter, Raghu Kiran Ganti, and Mayank Mishra. 2024. Enhancing training efficiency using packing with flash attention. *Preprint*, arXiv:2407.09105.

Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica. 2023. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*.

Andreas Köpf, Yannic Kilcher, Dimitri von Rütte, Sotiris Anagnostidis, Zhi-Rui Tam, Keith Stevens, Abdullah Barhoum, Nguyen Minh Duc, Oliver Stanley, Richárd Nagyfi, Shahul ES, Sameer Suri, David Glushkov, Arnav Dantuluri, Andrew Maguire, Christoph Schuhmann, Huu Nguyen, and Alexander Mattick. 2023. Openassistant conversations – democratizing large language model alignment. *Preprint*, arXiv:2304.07327.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason E Weston, and Mike Lewis. 2024. Self-alignment with instruction backtranslation. In *The Twelfth International Conference on Learning Representations*.

Sourab Mangrulkar, Sylvain Gugger, Lysandre Debut, Younes Belkada, Sayak Paul, and Benjamin Bossan. 2022. Peft: State-of-the-art parameter-efficient fine-tuning methods. https://github.com/huggingface/peft.

Yixin Ou, Ningyu Zhang, Honghao Gui, Ziwen Xu, Shuofei Qiao, Runnan Fang, Lei Li, Zhen Bi, Guozhou Zheng, and Huajun Chen. 2024. EasyInstruct: An easy-to-use instruction processing framework for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 94–106, Bangkok, Thailand. Association for Computational Linguistics.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John

Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, Alban Desmaison, Andreas Köpf, Edward Yang, Zach DeVito, Martin Raison, Alykhan Tejani, Sasank Chilamkurthy, Benoit Steiner, Lu Fang, Junjie Bai, and Soumith Chintala. 2019. *PyTorch: an imperative style, high-performance deep learning library*. Curran Associates Inc., Red Hook, NY, USA.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2023. Direct preference optimization: Your language model is secretly a reward model. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Samyam Rajbhandari, Jeff Rasley, Olatunji Ruwase, and Yuxiong He. 2019. Zero: Memory optimization towards training A trillion parameter models. *CoRR*, abs/1910.02054.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, YK Li, Y Wu, et al. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Mohammad Shoeybi, Mostofa Patwary, Raul Puri, Patrick LeGresley, Jared Casper, and Bryan Catanzaro. 2019. Megatron-lm: Training multi-billion parameter language models using model parallelism. *CoRR*, abs/1909.08053.

Charlie Snell, Dan Klein, and Ruiqi Zhong. 2022. Learning by distilling context. *arXiv preprint arXiv:2209.15189*.

Abhishek Thakur. 2024. AutoTrain: No-code training for state-of-the-art models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 419–423, Miami, Florida, USA. Association for Computational Linguistics.

Vijay Viswanathan, Chenyang Zhao, Amanda Bertsch, Tongshuang Wu, and Graham Neubig. 2023. Prompt2Model: Generating deployable models from natural language instructions. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 413–421, Singapore. Association for Computational Linguistics.

Leandro von Werra, Younes Belkada, Lewis Tunstall, Edward Beeching, Tristan Thrush, Nathan Lambert, Shengyi Huang, Kashif Rasul, and Quentin Gallouédec. 2020. Trl: Transformer reinforcement learning. `https://github.com/huggingface/trl`.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-instruct: Aligning language models with self-generated instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 13484–13508, Toronto, Canada. Association for Computational Linguistics.

Jason Wei, Maarten Bosma, Vincent Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M. Dai, and Quoc V Le. 2022. Finetuned language models are zero-shot learners. In *International Conference on Learning Representations*.

Hao Xiang, Bowen Yu, Hongyu Lin, Keming Lu, Yaojie Lu, Xianpei Han, Le Sun, Jingren Zhou, and Junyang Lin. 2024. Aligning large language models via self-steering optimization. *arXiv preprint arXiv:2410.17131*.

Zhangchen Xu, Fengqing Jiang, Luyao Niu, Yuntian Deng, Radha Poovendran, Yejin Choi, and Bill Yuchen Lin. 2025. Magpie: Alignment data synthesis from scratch by prompting aligned LLMs with nothing. In *The Thirteenth International Conference on Learning Representations*.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2023. Rlcd: Reinforcement learning from contrastive distillation for language model alignment. *arXiv preprint arXiv:2307.12950*.

Weizhe Yuan, Richard Yuanzhe Pang, Kyunghyun Cho, Xian Li, Sainbayar Sukhbaatar, Jing Xu, and Jason Weston. 2024. Self-rewarding language models. *Preprint*, arXiv:2401.10020.

Yaowei Zheng, Richong Zhang, Junhao Zhang, Yanhan Ye, and Zheyan Luo. 2024. LlamaFactory: Unified efficient fine-tuning of 100+ language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 3: System Demonstrations)*, pages 400–410, Bangkok, Thailand. Association for Computational Linguistics.

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023. Lima: Less is more for alignment. *Preprint*, arXiv:2305.11206.

A Basic Alignment Techniques

As an all-in-one toolkit for alignment, in this chapter we present a series of practices for aligning models ranging from 7B to 72B with basic fine-tuning methods using AUTOALIGN. Throughout the process, we employ UltraChat (Ding et al., 2023) and UltraFeedback (Cui et al., 2023), which are widely adopted in academia as training data. This series of experiments also provides usable baselines for the academic community.

Model	MT-Bench (EN)	MATH (EN)	GSM-8K (EN)	HumanEval (EN)	MBPP (EN)	HumanEval-CN(ZH)	MBPP-CN(ZH)	MMLU (EN)	GPQA (EN)	CMMLU (ZH)	C-Eval (ZH)	BBH (EN)
Llama-3-8B$_{\text{UltraChat}}$	5.41	13.58	59.89	20.12	36.00	28.05	31.40	62.92	29.29	48.38	46.89	59.13
Llama-3-8B$_{\text{UltraChat_UltraFeedback}}$	6.17	14.64	53.53	29.27	35.40	29.88	28.40	63.50	30.81	48.47	48.21	57.61
Llama3-70b$_{\text{UltraChat}}$	6.29	31.80	82.34	20.73	45.40	20.12	42.40	75.30	26.77	64.25	62.16	79.60
Llama3-70B$_{\text{UltraChat_UltraFeedback}}$	6.49	32.70	73.69	31.71	42.40	17.68	44.40	76.45	27.27	64.49	63.31	81.64
Qwen2-7B$_{\text{UltraChat}}$	5.93	40.34	81.96	46.34	37.60	40.85	36.00	69.80	33.84	81.83	82.15	61.30
Qwen2-7B$_{\text{UltraChat_UltraFeedback}}$	6.61	42.48	79.38	49.39	39.80	48.17	38.60	70.11	31.31	82.20	82.66	61.58
Qwen2-72B$_{\text{UltraChat}}$	6.79	50.20	89.08	45.12	47.40	31.71	45.00	81.13	28.28	89.62	90.25	79.61
Qwen2-72B$_{\text{UltraChat_UltraFeedback}}$	6.94	52.70	88.55	59.15	46.60	48.78	45.40	82.01	32.83	88.49	90.24	81.17
Qwen2.5-7B$_{\text{Infinite_9M}}$	6.85	39.44	84.08	71.95	58.40	64.02	55.20	74.51	37.88	78.79	80.23	71.03

Table 5: Performance Reference with the standard alignment training datasets UltraChat and UltraFeedback.

			Anthropic-Helpful	OpenAI WebGPT	OpenAI Summ.	Stanford SHP	Reward-Bench
Llama-2-13B	UltraFeedback	Official Report	66.7	65.1	66.8	68.4	67.6
Llama-2-13B	UltraFeedback$_{\text{Mixture}}$	Official Report	71.0	65.2	74.0	73.7	—
Llama-3-8B	UltraFeedback$_{\text{Binary}}$	—	62.56	—	69.67	67.41	73.68

Table 6: Reward Model Training Performance with AUTOALIGN.

A.1 Supervised Finetuning

Supervised Finetuning (SFT) is a fundamental technique in Post-Training where models are trained on specific datasets to improve their capabilities on targeted tasks.

SFT helps models better align with human preferences and expectations by learning from high-quality demonstrations of desired outputs. This process typically involves training on instruction-response pairs to teach the model how to follow user instructions effectively.

As shown in the Table 5, models like Llama-3-8B and Qwen2-7B benefit from SFT with UltraChat data. We also train Qwen2.5-7B on very large scale SFT data (Infinite-Instruct$_{9M}$) to further demonstrate the potential of SFT, the resulting model show great improvement on code and math abilities.

Rejection-Sampling Finetuning As a variant of Supervised Finetuning. AUTOALIGN also supports customize rules to iteratively filter training data, i.e., Rejection-Sampling Finetuning.

A.2 Reinforcement Learning

Direct Preference Optimization Direct Preference Optimization (DPO) is an efficient alternative to traditional reinforcement learning methods to further improve the Supervised Finetuned models. It directly optimizes model outputs based on human preferences without explicitly training a reward model.

In Table 5, we see models with UltraFeedback suffix underwent DPO training with AutoAlign. For example, Qwen2-7B shows improvement from 5.93 to 6.61 on MT-Bench after DPO training, demonstrating how this technique can effectively enhance model alignment with human preferences in a second stage, bypassing the complexity of RLHF's multi-stage process.

Phase	Global Batch Size	Learning Rate
IFT	64	5×10^{-6}
EFT	64	2×10^{-7}
DPO Iteration 1	64	5.5×10^{-8}
DPO Iteration 2	64	3×10^{-8}

Table 7: Training hyperparameters in Self-Rewarding

Group Relative Policy Optimization Group Relative Policy Optimization (GRPO) (Shao et al., 2024) is a policy gradient algorithm that eliminates the need for a value function model and instead uses the average reward of multiple sampled outputs from the same problem as a baseline to estimate the advantage function, thereby significantly reducing the memory and computational overhead of the PPO algorithm. This method maintains training effectiveness while avoiding the complexity of training an additional value network, making it particularly suitable for model optimization in mathematical reasoning tasks.

Reward Modeling Reward modeling involves training a separate model to evaluate the quality of responses, which can then be used to guide the auto alignment process. Table 6 shows reward model training performance with AUTOALIGN, where the Llama-3-8B model trained on UltraFeedback$_{\text{Binary}}$ achieves 73.68% accuracy on Reward-Bench. The

reward models trained with AUTOALIGN can be further used for iteratively filtering the on-policy data for policy iteration.

B Implementation Details

B.1 Details in Self-Rewarding Reproduction

Hyperparameters of Training Table 7 shows the hyperparameters of each training phase in self-rewarding.