

Baichuan Alignment Technical Report

Mingan Lin¹, Fan Yang¹, Yanjun Shen¹, Haoze Sun¹, Tianpeng Li¹, Tao Zhang¹, Chenzheng Zhu¹
 Tao Zhang¹, Miao Zheng¹, Xu Li¹, Yijie Zhou¹, Mingyang Chen¹, Yanzhao Qin², Youquan Li²
 Hao Liang², Fei Li¹, Yadong Li¹, Mang Wang¹, Guosheng Dong¹, Kun Fang¹, Jianhua Xu¹
 Bin Cui², Wentao Zhang², Zenan Zhou¹♣, Weipeng Chen¹

¹Baichuan Inc.

²Peking University

Abstract

We introduce Baichuan Alignment, a detailed analysis of the alignment techniques employed in the Baichuan series of models. This represents the industry's first comprehensive account of alignment methodologies, offering valuable insights for advancing AI research. We investigate the critical components that enhance model performance during the alignment process, including optimization methods, data strategies, capability enhancements, and evaluation processes. The process spans three key stages: Prompt Augmentation System (PAS), Supervised Fine-Tuning (SFT), and Preference Alignment. The problems encountered, the solutions applied, and the improvements made are thoroughly recorded.

Through comparisons across well-established benchmarks, we highlight the technological advancements enabled by Baichuan Alignment. Baichuan-Instruct is an internal model, while Qwen2-Nova-72B and Llama3-PBM-Nova-70B are instruct versions of the Qwen2-72B and Llama-3-70B base models, optimized through Baichuan Alignment. Baichuan-Instruct demonstrates significant improvements in core capabilities, with user experience gains ranging from 17% to 28%, and performs exceptionally well on specialized benchmarks. In open-source benchmark evaluations, both Qwen2-Nova-72B and Llama3-PBM-Nova-70B consistently outperform their respective official instruct versions across nearly all datasets. This report aims to clarify the key technologies behind the alignment process, fostering a deeper understanding within the community. Llama3-PBM-Nova-70B model is available at <https://huggingface.co/PKU-Baichuan-MLSystemLab/Llama3-PBM-Nova-70B>.

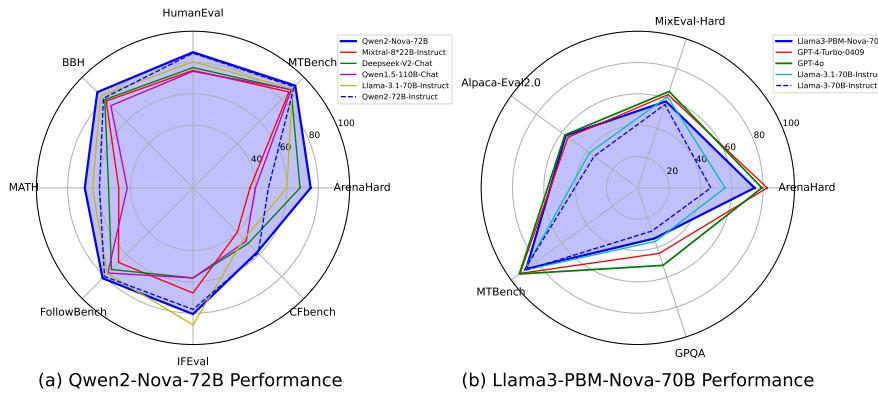


Figure 1: Performance Comparison of Qwen2-Nova-72B and Llama3-PBM-Nova-70B with Others

♣Project lead and Corresponding author: zhouzenan@baichuan-inc.com

Contents

| | | |
|----------|--|-----------|
| 1 | Introduction | 3 |
| 2 | Optimization | 4 |
| 2.1 | Training | 4 |
| 2.2 | Efficient Training | 5 |
| 2.3 | Prompt Augmentation | 5 |
| 2.4 | Model Merging | 6 |
| 3 | Data | 7 |
| 3.1 | Prompt Selection | 7 |
| 3.1.1 | Prompt System and Classification | 7 |
| 3.1.2 | Prompt Diversity | 8 |
| 3.1.3 | Prompt Quality | 8 |
| 3.2 | Response Construction | 9 |
| 3.3 | Preference Data | 10 |
| 4 | Key Ability | 10 |
| 4.1 | Instruction Following | 10 |
| 4.2 | Math | 12 |
| 4.3 | Reasoning | 12 |
| 4.4 | Code | 13 |
| 4.5 | Tool-using | 14 |
| 4.6 | Prompt Augmentation System | 15 |
| 5 | Evaluation | 15 |
| 5.1 | User Experience Evaluation | 16 |
| 5.1.1 | Evaluation Criteria | 16 |
| 5.1.2 | Result and Discussion | 16 |
| 5.2 | Open-Source Benchmarks | 16 |
| 5.2.1 | Benchmarks | 17 |
| 5.2.2 | Major Results and Discussion | 17 |
| 5.3 | Key Ability Evaluation | 18 |
| 5.3.1 | CFBench | 18 |
| 5.3.2 | SysBench | 19 |
| 5.3.3 | FB-Bench | 20 |
| 6 | Conclusion | 21 |

1 Introduction

In recent years, Large Language Models (LLMs) have achieved significant breakthroughs [67, 23, 69, 10, 18, 1, 3, 4, 81, 79], showing early signs of advancing toward Artificial General Intelligence (AGI). Through the mechanism of next-token prediction, LLMs undergo self-supervised pre-training on vast datasets, thereby acquiring a diverse spectrum of capabilities that empower them to perform an array of tasks, such as text continuation, summarization, and creative composition. Further advancements in alignment methodologies, such as Prompt Augmentation Systems (**PAS**) [103], Supervised Fine-Tuning (**SFT**), and preference modeling [65], have become important to the evolution of LLMs. These developments have enhanced the models' ability to comprehend user intentions and adhere to instructions, thereby enhancing their conversational skills and adaptability to complex real-world scenarios. Despite its critical importance, a comprehensive understanding of alignment remains largely inaccessible to the broader public [5, 95, 84, 19, 98, 8, 55, 31, 37, 94, 81, 1]. The broad and intricate nature of alignment results in fragmented research efforts, which often provide only a narrow insight into specialized areas, making it challenging to present a holistic perspective of the alignment landscape. Additionally, alignment is frequently regarded as a proprietary cornerstone in the LLMs training, shrouded in corporate confidentiality. To foster advancement within the LLM community, we present a comprehensive and systematic exposition of Baichuan Alignment, featuring a suite of advanced and practical alignment techniques.

Baichuan Alignment comprises three critical phases: Prompt Augmentation Systems (**PAS**), Supervised Fine-Tuning (**SFT**) and Preference Alignment. The PAS stage aims to transform user queries into instructions that are more comprehensible and actionable for LLMs through automated Prompt Engineering (PE) techniques. The SFT stage equips LLMs with the ability to engage in dialogue and handle complex tasks using a large corpus of high-quality and diverse data. The Preference Alignment further aligns the LLMs with human values and preferences. This report primarily focuses on four aspects: optimization, data, key capability enhancement, and system evaluation, which are critical elements of Baichuan Alignment. The optimization ensures the effectiveness and efficiency of LLM training, and Section 2 provides a detailed discussion on methods for training, prompt argumentation, and model merging, which collectively accelerate and improve model performance. In Section 3, we emphasize the importance of data in alignment, with a focus on prompt selection, response construction, and preference data. Section 4 outlines the challenges encountered in enhancing core capabilities on the path to AGI, detailing the specific approaches and insights gained through Baichuan Alignment. Finally, Section 5 presents a comprehensive evaluation of Baichuan Alignment, focusing on user experience, open-source benchmarks, and specific capability assessments. This evaluation is crucial for assessing model capabilities and guiding iterative improvements, particularly through user evaluations and third-party product assessments that align closely with user experience. Additionally, we introduce tailored benchmarks, including CFBench [99], SysBench [66], and FB-Bench [52], designed to address current challenges and practical needs in LLM applications.

We conducted a systematic evaluation of multiple models after Baichuan Alignment from various perspectives. Among these, Baichuan-Instruct serves as our internal model, while Qwen2-Nova-72B and Llama3-PBM-Nova-70B are instruct versions derived from the Qwen2-72B [95] and Llama-3-70B [27] base models, respectively, fine-tuned using Baichuan Alignment techniques. User experience assessments reveal that Baichuan-Instruct exhibits significant improvements across several core capabilities, with enhancements ranging from 17% to 28%. Notably, there are marked improvements in mathematics and reasoning, with increases of 28% and 23%, respectively. In terms of open-source benchmarks, Qwen2-Nova-72B demonstrates substantial advancements over its official instruct version, Qwen2-72B-Instruct, across multiple leaderboards. Similarly, Llama3-PBM-Nova-70B shows significant improvements compared to Llama-3-70B-Instruct, particularly achieving a 60% relative increase (from 46.6 to 74.5) on the ArenaHard benchmark. On specific benchmarks, Baichuan-Instruct ranks competitively in core competencies on CFBench, SysBench, and FB-Bench, compared to the leading models currently available. This multidimensional, comprehensive, and in-depth evaluation underscores the superiority of Baichuan Alignment technology and its applicability across different foundational models. As a crucial step toward AGI, we are publicly sharing the challenges encountered during the Baichuan Alignment process, the solutions devised, and some in-depth insights. This disclosure aims to provide the community with new perspectives or lessons learned from failures, thereby fostering discussions on alignment and making a meaningful contribution to the advancement toward AGI.

2 Optimization

2.1 Training

SFT During supervised fine-tuning, we optimize models using a learning rate of 1×10^{-5} and conduct training over 2 to 6 epochs for models with various size. For all training sessions, we employ sample packing, which will be discussed below. Additionally, we apply weight decay to prevent overfitting.

Reward In many practice [64, 6, 55], a reward function is estimated based on the preference data using the Bradley-Terry [9] model. However, the Bradley-Terry model has some limitations, including a tendency to overfit the data. This can occur even before completing a single epoch of training despite having a high-quality datasets. Besides, the reward model only ensure the relative order of the reward score for different responses, not their absolute feeling. Eg: the fitted reward score difference of $S_{perfect} - S_{bad1}$ may less than $S_{bad1} - S_{bad2}$, which encourage the model to find a shortcut to hack the reward. To mitigate such impacts, we add a point wise MSE loss to let the model fit the normalized absolute score which annotated from Section 3.3. Thus the reward model will minimize the following objective:

$$\begin{aligned} \mathcal{L}_\theta = E_{(x,y_w,y_l)} & \left[-\log (\sigma(r_\theta(x, y_w) - r_\theta(x, y_l))) \right. \\ & \left. + \alpha ((r_\theta(x, y_w) - \hat{r}_x^{y_w})^2 + (r_\theta(x, y_w) - \hat{r}_x^{y_l})^2) \right] \end{aligned} \quad (1)$$

where the α is a adjustable coefficient, $\hat{r}_x^{y_w}$ and $\hat{r}_x^{y_l}$ are the normalized annotated absolute score for the chosen and rejected data. Note that in our preference dataset, we also mixed some dataset without absolute score, like open source hh-rlhf[7] dataset, SHP[28] dataset. For data in these dataset, the α is set to 0. With this objective, we find the fitted reward model is more robust. The RM model is trained for 1 to 2 epochs, depending on its size.

Reinforcement Learning During the course of reinforcement learning, we conducted experiments on both PPO[73] and GRPO [74] to further enhance our model. When applying these two approaches, we made few modifications: When doing the Reinforcement Learning to merged models(described in section 2.4), using cross-entropy loss on SFT dataset as PTX loss would degrade the model into an SFT+reinforcement model. Therefore, during the reinforcement training, we use the KL divergence between the policy model and the original model as PTX loss. Furthermore, when calculating KL divergence of each token, we first select the indices corresponding to the Top500 logits from the reference model. For these Top500 indices, we separately computed the normalized log probabilities from both the policy model and the reference model. We then applied the standard KL divergence formula between the two sets of log probabilities, rather than using the simplified KL divergence version [72], to ensure the KL remained non-negative and as accurate as possible. In GRPO, the n_sample for each prompt is set to 3 and we did not remove the KL term in token-level reward decomposition. The optimization objective of the Reinforcement Learning would described as below:

$$\begin{aligned} \text{objective}(\theta) = & \mathbb{E}_{(x,y) \sim D_{\pi_\theta^{RL}}} [r_\theta(x, y) - \beta \mathbb{KL}(\pi_\theta(y|x) || \pi_{ref}(y|x))] \\ & + \gamma \mathbb{KL}_{(x,y) \sim D_{SFT}} [\pi_\theta(y|x) || \pi_{ref}(y|x)] \end{aligned} \quad (2)$$

During the training process, the model would be save and evaluated after every certain training steps. We establish corresponding test sets for various task categories to monitor the over-fitting or under-fitting of each category. This allows us to adjust the prompts proportion for each category in the training data, thereby enhancing the overall performance of the model.

In our experiment, we found that even without the critic model, GRPO can still achieve comparable results with PPO. The evaluation benchmark(described in Section 5.2.1) difference between GRPO and PPO $\Delta_{GRPO-PPO}$ is +3.7% for FollowBench, +1.0% for CFBench PSR Full and -0.50% for SysBench. On the other hand, GRPO can save nearly half of the training resources compared to PPO. Additionally, it achieves better performance than direct optimization methods like DPO[68] and KTO[29], while requiring only a marginal increase in training resources. Therefore, the GRPO is chosen as our Reinforcement Learning method.

2.2 Efficient Training

Packing To address training inefficiencies stemming from the varying lengths of samples, we consolidated multiple short samples into a single long sample, significantly reducing the number of padding tokens required [80]. Traditional packing is performed at the sample level and typically implemented on the `attention_map` of Flash Attention v1. In our approach, we leveraged the `cu_seqlens` argument in Flash Attention v2 [22], which allows for mask-free variable sequence lengths. This feature effectively isolates attention across different samples, preventing contextual contamination between them and enabling plug-and-play equivalent training. Our experiments demonstrated that this method of packing increased the effective token utilization rate within a batch from 10% to 98%, achieving nearly a tenfold improvement in efficiency without any loss in performance.

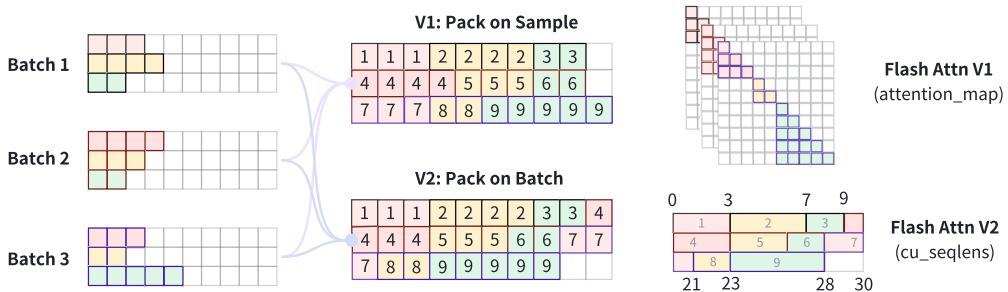


Figure 2: Difference between packing on sample and packing on batch.

Multi-layer Gradient Checkpointing Conventional practical implementation of gradient checkpointing typically involves setting a checkpoint for each decoding layer for LLMs [16]. However, in models with a very large number of layers, such as those with over 80 layers and more than 70 billion parameters, this setup is far from the most memory-efficient configuration. Gradient checkpointing is essentially a trade-off between time and memory. Ideally, the configuration should allow the GPU memory to accommodate a sequence length that fully utilizes the computational power, making it the most cost-effective setup. In practice, this parameter requires experimental tuning. Therefore, we use the multi-layer gradient checkpointing that merges multiple decoding layers to achieve better memory control. Assuming that during the forward pass of the model, each decoder layer has n activations, then during the backward pass, the amount of storage required for gradient checkpointing is $hidden_size \times n + hidden_size \times num_layer$. If k layers are merged before checkpointing, the required storage becomes $k \times hidden_size \times n + hidden_size \times \frac{num_layer}{k}$. This implies that when $k = \sqrt{\frac{num_layer}{n}}$, the memory usage is minimized. It is important to note that the value of n and the calculation of the optimal k may vary depending on the specific architecture. In our practice, this optimization can reduce the minimum number of GPUs required to train a model more than 70 billion with a sequence length of 16K from 128 GPUs to 40 GPUs.

Sequence Parallel Following DeepSpeed-Ulysses [39], we adapt the sequence parallelism method to our training framework, and achieve efficient and scalable training for large language models with extremely long sequence lengths. This makes it more suitable for high-frequency experiments and scenarios with limited total resources.

2.3 Prompt Augmentation

Variations in application contexts demand tailored response paradigms from large-scale language models. For professional inquiries, the model should adopt an authoritative and informative tone, while for consumer interactions, it should strive for an empathetic and engaging tone. In both cases, the model’s ability to adapt its style to the context is crucial. The model’s flexibility in adjusting its communication style is key to meeting the diverse needs of different user groups. Nonetheless, inconsistent response styles in training data may adversely affect the model’s performance. If the

data it is trained on lacks consistency, the model may struggle to develop a coherent and effective communication strategy. A feasible approach is to decouple the model's capabilities from the requirements for its response style. This allows for greater flexibility in optimizing and fine-tuning the model's functionality, while also enabling the customization of its response style according to varied needs or scenarios.

The effectiveness of large language models in various applications largely depends on the prompts' quality. There are already many designed prompts that can significantly enhance the performance of LLMs [43, 85, 88, 97]. However, these methods that rely on manual prompt engineering are far less scalable and have steep learning curves and significant time investment for consumer-side users. Therefore, it is essential to explore the development of mechanisms for automatic prompt engineering, which could significantly enhance the efficiency and effectiveness of AI interactions.

We designed a plug-and-play system to automatically generate prompt complementary, named Prompt Augmentation System (PAS). The pipeline, in general, generates corresponding complementary content based on the user's prompt, and then the prompt is supplemented with the content and input into the main model. PAS usually supplements the following aspects of content: 1) Requirements for product applications scenario responses. 2) The extension content based on user intent with non-mandatory tunes. 3) Response format constraints, like "the content should be logically clear, organized, and easy for users to understand."

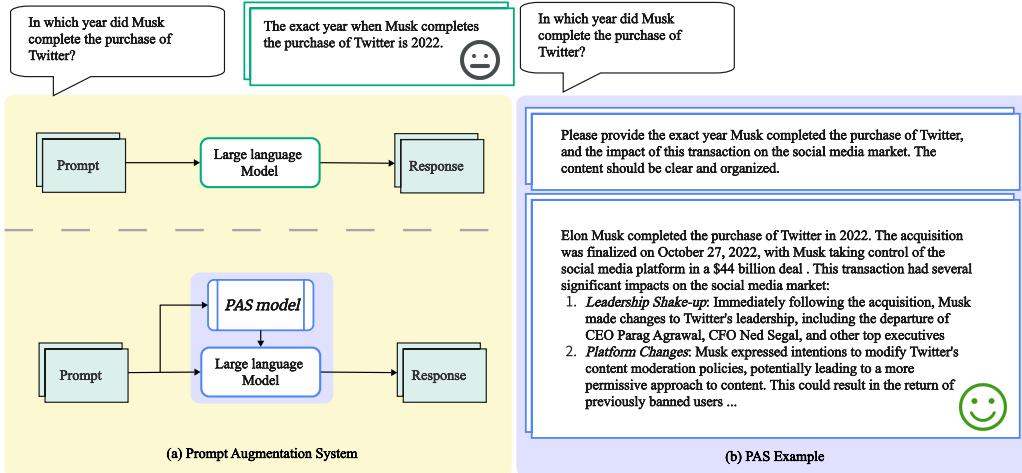


Figure 3: We present Prompt Augmentation System (PAS). (a) It takes user prompts, enhances them, and inputs the augmented prompts into LLMs. (b) PAS significantly improves responses across all categories in human evaluation.

Regarding Retrieval Augmented Generation (RAG) for LLMs, the extended prompt content needs to be within the scope of the search results; therefore, when generating supplementary prompts, the search keywords should be used as new constraints.

2.4 Model Merging

Model merging is an emerging technique in the field of artificial intelligence that involves combining the parameters of multiple models, each optimized for different tasks, to create a more flexible and universal model. Despite being a relatively nascent area of research, model merging is rapidly advancing and has already demonstrated its utility across various domains. A notable application of this technique is in the enhancement of foundation models. By merging models fine-tuned on different downstream tasks, the capabilities of large language models can be significantly augmented. For an in-depth exploration of model merging, we refer readers to a comprehensive survey of merging algorithms [96].

One of the challenges in model training is the proliferation of fine-tuned checkpoints, which result from different training data updates, hyperparameter settings, and regularization techniques. These variations often lead to divergent performance outcomes across different domains, a phenomenon

commonly referred to as the "seesaw effect", where improvements in one domain result in deteriorations in another. Model merging offers a promising solution to mitigate this effect by balancing the performance across diverse tasks.

In practice, we selected the best-performing models from different domains and applied various merging algorithms, including Linear, Task Arithmetic [38], and Model Stock [40], using the model merging toolkit, MergeKit [32]. Our experimental results indicate that the merged models typically achieved more balanced performance across the evaluated domains. Among the tested algorithms, Model Stock consistently delivered the best overall performance.

3 Data

Alignment data has been extensively validated as critical to the ultimate performance of LLMs, including both prompt response pairs and preference data [46, 90, 44, 26, 30]. Figure 4 delineates the comprehensive pipeline for constructing the Baichuan alignment dataset. Initially, we develop an prompt system and a classification model (Section 3.1.1), which form the foundational basis of the data flywheel. We then elaborate on the pivotal steps in data construction, focusing on prompt diversity (Section 3.1.2) and prompt quality (Section 3.1.3). Furthermore, we detail various techniques and best practices in generating responses (Section 3.2). Additionally, we provide insights derived from the process of constructing preference data (Section 3.3).

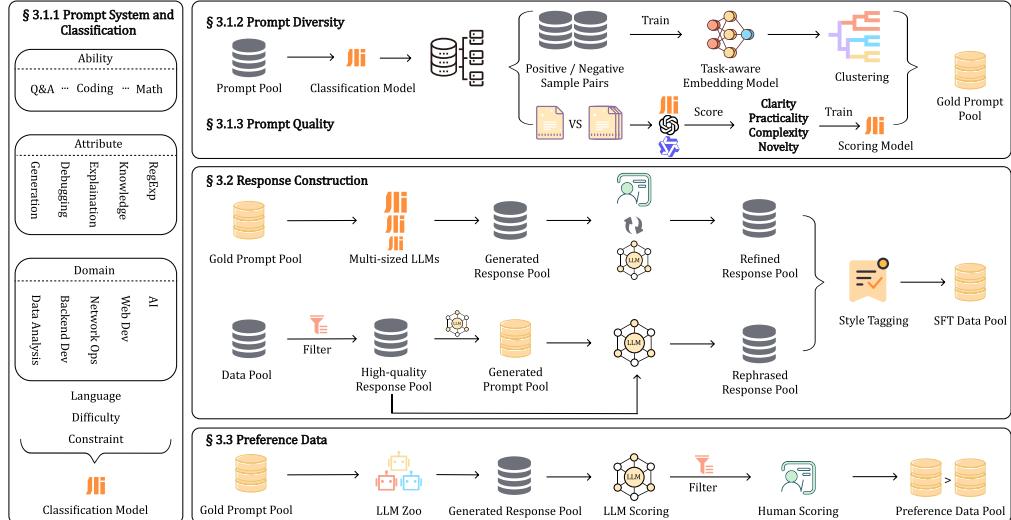


Figure 4: The Pipeline of Alignment Data Processes, including: Prompt System and Classification, Prompt Selection, and Construction of Response and Preference Data

3.1 Prompt Selection

3.1.1 Prompt System and Classification

The prompt system plays a essential role in data management, augmenting the breadth and balance of data coverage, and steering the reliability and direction of model performance iterations. Guided by taxonomy and statistical analysis, and supplemented by human expertise, we have developed a multi-dimensional, multi-granularity prompt labeling system through numerous iterations and refinements. Subsequently, an automated labeling classification model is trained and deployed across various facets of data production.

Prompt System We curated a diverse initial prompt pool, incorporating real requests from human prompts and various open-source datasets. Leveraging a general model, preliminary labels were assigned to these prompt. Subsequently, an initial prompt labeling system was developed through techniques such as synonym merging, long-tail pruning, and hierarchical association. Through

iterative refinements and expansions by human annotators, the final version of the labeling system and the enriched label dataset were meticulously established.

Notably, our system is meticulously structured around six primary dimensions: ability, attributes, domains, language, difficulty and prompt constraints. These dimensions, featuring both interwoven and hierarchical structures, enable the creation of thousands of combinatory types. Ability encompass the skills required by the LLM to complete tasks, including knowledge-based Q&A, text generation, code programming, and logical reasoning. Attributes provide unique contextual information linked to capabilities, such as literary writing and practical writing for text generation. Domains connect the real world to the LLM, imbuing it with relevance and vitality, with common domains including IT, history, and humanities. Language defines the medium of expression, categorized into Chinese, foreign languages, and programming languages, with further subdivisions such as Simplified Chinese, Traditional Chinese, Classical Chinese, English, French, Python, and C++. Difficulty indicates the complexity of prompt, classified into easy, intermediate, and difficult levels. Prompts constraints highlight the importance of constraints within prompts, categorized by the number of constraints into unconstrained, simple constraints, and strong constraints. This integrated framework ensures a comprehensive and adaptable system, adept at addressing a wide array of instructional scenarios.

Classification Model Under the guidance of the aforementioned prompt system, we initially utilized advanced LLMs to classify prompts. This process was refined through methods such as voting and manual verification of low-confidence samples, resulting in the construction of a training set comprising tens of thousands of examples. Subsequently, we fine-tuned a specialized automatic prompt labeling model based on Baichuan2-13B [94] and the label data, achieving a 90% accuracy rate on the evaluation set through prompt-based techniques. In contrast, a fine-tuned BERT [24] model only reached an accuracy of 81%. The integration of the prompt system with the automatic classification model enables efficient management and iteration of the prompt set. This includes evaluating prompt diversity and coverage, grouping for mining and optimization, and automatically matching prompts to specialized human annotators. Every new data point added to the training set undergoes this systematic process, ensuring consistent and comprehensive handling of all prompt-related tasks, as shown on the far left of Figure 4.

3.1.2 Prompt Diversity

Numerous studies have demonstrated that similar and repetitive prompts can adversely affect model performance, underscoring the critical importance of prompt diversity during the alignment phase [87, 89, 12]. In practical applications, prompts typically comprise both instructions and inputs, often formatted as task templates concatenated with contextual information. Current semantic-based dense embedding methods generally model the overall semantics but fail to adequately capture the repetitive representation of task-specific information [70, 91]. To address this challenge, we propose a task-aware embedding model that more precisely captures the nuanced differences between instructions, thereby facilitating the selection of a more diverse set of prompts.

As depicted in the upper right section of Figure 4, our primary innovation lies in extracting high-quality task-aware training data through a multi-granularity clustering approach. We begin with coarse-grained clustering, followed by fine-grained clustering within each category. Using the Longest Common Subsequence (LCS) algorithm and heuristic rules, from different fine-grained clusters, samples with similar task templates are identified as hard positive samples, while those without similar task templates are identified as hard negative samples. These samples are then used in contrastive learning with Triplet Loss to train a high-quality embedding model. Additionally, we incorporate hierarchical clustering principles by setting an incremental sequence of thresholds to perform layered clustering, thereby enhancing the robustness, efficiency, and stability of the algorithm. This combined approach allows us to achieve significantly superior results compared to the original algorithm, using only 50% of the original data volume.

3.1.3 Prompt Quality

High-quality prompts are instrumental in training models more efficiently to achieve superior performance [104, 93, 56]. However, existing prompt quality scoring methodologies exhibit certain deficiencies, as they either lack robustness across general datasets or fail to satisfy the personalized instruction screening requirements in specific contexts [34, 57, 53, 11, 13]. To address these issues,

we have developed a flexible and scalable automated prompt quality evaluation framework leveraging large language models (LLMs).

Training Data Drawing inspiration from the ArenaHard evaluation system [47], we utilize a pairwise LLM-based judging mechanism to construct training dataset. Specifically, we initially employ the classification model outlined in Section 3.1.1 to categorize instructions into 20 distinct buckets. From each bucket, we randomly select 30 data points to serve as anchors. Subsequently, other data within the bucket are paired with these anchors, and multiple LLMs are employed as the evaluator. The evaluation process considers four dimensions: Clarity, Practicality, Complexity, and Novelty, using a three levels scoring system for pairwise comparisons. The aggregated scoring results from multiple anchors are ultimately used to assign the final quality label to each example.

Performance and Effectiveness In consideration of performance and efficiency, we fine-tuned the Baichuan2-7B [94] base model using the aforementioned training data to develop the final prompt scoring model, Quality-7B. This model offers substantial advantages in terms of efficiency and cost compared to powerful LLMs such as GPT-4 [1]. Furthermore, through testing on a set of 200 evaluation samples, we have demonstrated that the Quality-7B model’s scoring accuracy significantly surpasses GPT-4.

3.2 Response Construction

Human Annotation We employ various scales of models and diverse generation strategies to sample multiple responses for the same prompt. These responses are automatically assigned to specialized annotators based on type labels for preference ranking, which greatly enhances annotation efficiency and quality, thereby significantly elevating the upper limit of data. Typically, we use a Reward Model or LLM-as-Judge to pre-filter responses, forming a response set with graded quality distinctions to collect a sufficiently rich preference order. When the best response fails to meet the established standards, we require annotators to modify the answers, thereby constructing a high-quality SFT and preference dataset.

Human-Machine Collaborative Annotation Large-scale annotation demonstration datasets are constrained by cost, time, and personnel. Therefore, we adopt a human-machine collaborative approach to enhance efficiency. For example, we use the critique from LLM as auxiliary information to improve the speed and quality of response modifications. Sometimes, we ask evaluators to compile a set of quality defects based on a sampled data subset, and then use LLM to perform defect mining and automatic rewriting. In practice, many useful techniques have been accumulated during the annotation process, and any attempts or methods that utilize LLM to enhance annotation efficiency are highly rewarded.

Instruction Back-Translation For tasks requiring expertise or creativity, it is unrealistic and uneconomical to expect annotators to always produce high-quality responses. To address this issue, inspired by [49], we use text quality models to filter high-quality texts from public sources across various fields and synthesize high-quality prompt-response pairs. We have collected a pool of high-quality copywriting, exemplary essays, and highly praised posts, and then back-translate them to generate corresponding creative instructions.

Personalization In the context of multiple correct responses to the same prompt, individuals may exhibit divergent preferences: some favor concise and direct answers, while others prefer detailed and structured responses. By incorporating style descriptions corresponding to different response styles into the prompts, we mitigate style preference conflicts and enhance the model’s adaptability to switch between styles. Additionally, responses often include refusals due to constraints related to values and hallucinations. Prompts concerning safety, timeliness, and model functionality can easily lead to excessive refusals, significantly reducing usability and harming user experience. We address this issue by strictly controlling the proportion of refusals and incorporating refusal constraints into the system message and prompt.

3.3 Preference Data

A dataset with high data quality and good data diversity is important not only for SFT dataset, but also for RLHF preference dataset. A data filter pipeline similar to Section 3.1 is performed to get the high quality and high diversity prompt collections with their labels for the preference dataset. In addition, we filter prompts with simple, time-sensitive meaningless or out-of-ability tasks using the classified labels. We also only keep the prompts with Chinese or English languages due to our annotators language limitations.

For the data annotation, we refer and improve the process pipeline of Llama 2 [82]. We use our Top 3 most advanced models to sample 5 responses for each prompt. The generation setup of each model is $Temperature = 1$, $TopP = 0.99$, $TopK = 50$. To further increase the data diversity and filter the prompts which model already answers well, a rouge[54] rule based filter is used to roughly filter the similar responses. For different set of tasks, we developed several LLM-as-Judge[102] prompts named AutoRator to evaluate the judged score of the response. In summary, there are three types of the AutoRator used in the judge pipeline: 1) Absolute score: Judge the absolute evaluate score for tasks with open responses like Writing, Open QA and etc. 2) Pair comparison: Judge if the two response have the similar thought chain and same result. For tasks which have specific answers like Math, Reasoning. 3) Golden Answer: Judge the absolute evaluate score with respect to the golden answer for datasets which have golden answers. After the AutoRator evaluation, prompts with all high-scoring responses or very similar scoring responses are removed, and the remaining data is sent to annotators. For some math and reasoning data with golden answers, we find the LLM-as-Judge score is good enough and directly use them.

When doing the annotation, for each group of response, we not only ask for annotators to rate the their preferred order, but also the absolute feeling score about the helpfulness, writing fluency and safety. To ensure a good annotation quality and for possible future use, we ask the annotators to highlight the erroneous part and add a note about the error if a response has factual errors or logical errors. If all of responses of a prompts get low absolute annotation score, it would be send to the response modify process described in Section 3.2.

Due to the majority of our annotated data is in Chinese language, we have incorporated additional open-source datasets in other languages to enhance the multilingual capabilities. These datasets include hh-rlhf[7], Helpsteer2[86], UltraFeedback[21] and SHP[28].

4 Key Ability

4.1 Instruction Following

The ability to following instruction is a critical capability of any advanced Large Language Model (LLM), particularly when confronted with the intricacies of real-world applications. Baichuan Alignment has implemented several key enhancements to bolster this crucial capability, as depicted in Figure 5. These primarily include the construction of complex system messages, expansion of instruction constraints, response reversal, and textbook learning.

System Message System message is a set of special instructions located at the beginning of multi-turn conversations, pre-setting the role, background, approach or output format of the model to align with the user-defined objectives [83, 45, 58, 62]. We enhance the understanding and adherence capabilities of LLMs by collecting and constructing a large-scale, high-quality dataset of system messages (see Figure 5 (a) for details). Initially, we collect a substantial amount of preliminary system message components. On one hand, we gather direct system messages from real user logs and open-source data, further decomposing them into task descriptions, workflows, output specifications, constraints, and initial statements. On the other hand, we extract samples with complex constraints and requirements from extensive multi-turn dialogues, utilizing LLMs to extract and synthesize similar components. Subsequently, we employ clustering and filtering techniques to obtain a diverse array of constraint descriptions and other component pools. We seamlessly integrate the original prompts with appropriate constraints to create constraint-rich prompts. Simultaneously, we meticulously select the remaining components of the system messages and further embed the constraint prompts into suitable positions, accompanied by evolutionary learning and constraint rationalization steps. This process is iterated multiple times, culminating in a final filtering step to eliminate unreasonable

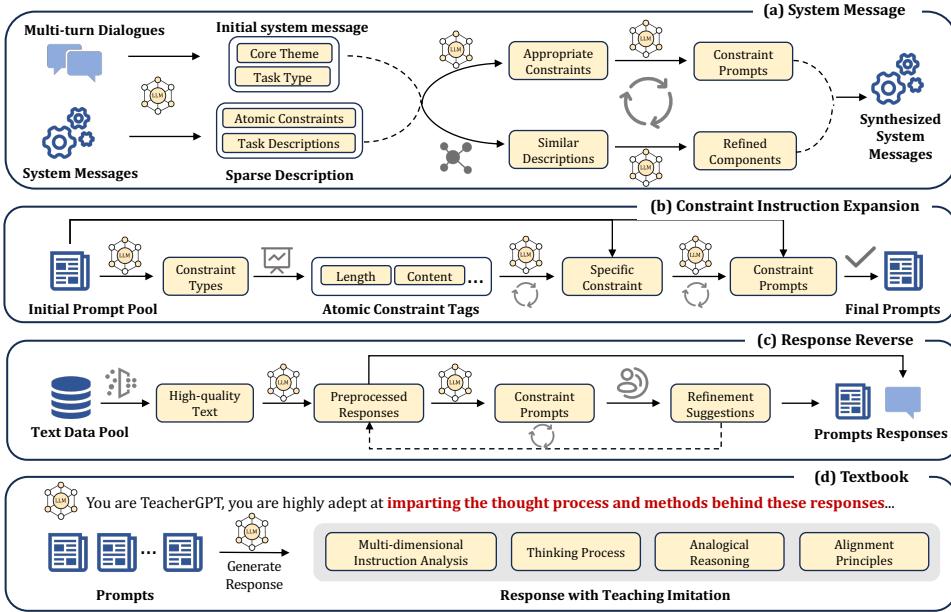


Figure 5: Overview of Instruction-Following Optimization, including: System Message, Constraint Expansion, Response Reversal, and Textbook Techniques.

system messages, resulting in a comprehensive system prompt database. Through these steps, we ensure that the system messages cover a diverse range of task types, such as role-playing, function call, and security. We also ensure the tight integration of constraint descriptions and task descriptions, as well as the overall high quality of the data.

Constraint Instruction Expansion The construction of complex instructions through the addition of constraints has been well-documented as crucial for enhancing instruction-following capabilities [76, 25, 33, 92]. To ensure the quality and coverage of constraint-based instruction data, we employ a series of steps and methodologies, including the development of a complex constraint system, collaborative iteration with multiple LLMs, and independent validation, as illustrated in Figure 5 (b). Drawing on previous work[99], we utilize LLMs to decompose, cluster, and synthesize over 30 types of constraints from a large pool of prompts. Through sampling, we verify that the coverage of constraint types in random instructions reaches up to 90%, and obtain rich descriptions corresponding to each type of constraint. Subsequently, we employ multiple LLMs to increase the complexity of constraints in selected original prompts. This process involves detailed prompt engineering guidance to ensure LLMs select constraint descriptions that match the prompts, focusing on the rationality of constraint types, the flexibility of constraint expression, and filtering based on prompt quality scores. Finally, we use a separate LLM as a validator to select high-quality, constraint-rich instructions.

Response Reversal The alignment of complex instructions poses two key challenges: (1) the scarcity and lack of diversity in high-quality prompts containing multiple constraints, and (2) the difficulty and high cost associated with producing responses that adhere precisely to these prompts. Nevertheless, there is an abundance of high-quality natural language text that can be utilized by leveraging the inherent attributes of these texts as constraints for generating corresponding instructions. This approach facilitates the large-scale creation of rich and high-quality prompt-response pairs with multiple constraints. To address this, we have developed a response reversal pipeline, as illustrated in Figure 5 (c). The detailed process is as follows: First, we perform quality filtering by employing a quality scoring model to extract high-quality text from knowledge-rich passages in web text. Next, using this high-quality text as the response and tasks a LLM with generating the corresponding prompt, incorporating the inherent attributes of the response as constraints for the prompt. Finally, the LLM inspector evaluates the quality of the generated prompts, assigns scores, and provides

suggestions for improvement. This process is iterated until all constraints present in the response are fully encapsulated within the prompt.

Textbook While simple imitation learning can efficiently acquire relatively straightforward instructions, it often results in only a superficial understanding when applied to complex instructions. To address the issue where models may know the outcome but lack comprehension of the underlying reasoning, we propose a method of instructional imitation. By explicitly focusing on learning the intent of the instruction, the thought process, analogical reasoning, and alignment principles, we can significantly enhance the efficiency of instruction learning and improve the model’s generalization and alignment capabilities. Specifically, for complex instructions, we leverage LLMs to generate responses that are highly aligned with the prompts, as illustrated in Figure 5 (d). For example, when tasked with understanding an instruction, the model might be prompted to “explain how to comprehend the instruction, analyzing its details and challenges from multiple dimensions such as professionalism, complexity, and output constraints.” Alternatively, LLMs can be guided to articulate the thought process behind a solution, such as “elaborate on the reasoning process behind the solution, including relevant domain knowledge, reasoning methods, and common pitfalls, particularly focusing on erroneous reasoning paths and easily overlooked output constraints like format, style, tone, length, and word count limitations.” Additionally, prompts like “you need to learn the reasoning techniques and background knowledge mining from the given examples. For instance: xx” can be fed to the LLM to endow it with the ability to generalize and adapt to different task scenarios. Some alignment principles are also taught, such as “avoid using technical terms or complex expressions whenever possible” and “consider the user’s emotional state,” which are common response techniques. This approach ensures a deeper understanding and more robust application of complex instructions, thereby enhancing the overall capability of the model to generalize and align with diverse instructional requirements.

4.2 Math

Prompt Collection During the prompt selection phase, a multi-level balanced approach is employed to maintain problem diversity. Specifically, problems are sampled evenly across elementary, middle, high school, and college levels. Furthermore, we use more fine-grained criteria, called “knowledge points”, where within each level, problems are selected to evenly cover over 1,000 different knowledge points. Priority is given to sampling problem-solving questions, as they generally include more comprehensive steps.

Response Generation In many datasets of math problem-solving, the provided answers often lack detailed steps. Manual annotation is a common method, however, some intermediate steps may be ignored. Nevertheless, these steps are important for model training. By leveraging large language models in combination with prompts and standard answers, we can generate detailed solutions that maintain a consistent style based on the reference answers. This approach synthesizes a substantial amount of math alignment data with detailed steps, significantly enhancing the model’s mathematical capabilities. Directly generated answers often struggle to match correct answers, so providing reference answers and steps helps the model enrich these into coherent reasoning chains. We find that reference answers significantly improve the quality of generated responses. However, since reference answers may contain errors, we instruct the model to consider them as having a 95% probability of correctness. This approach reduces incorrect inferences based on potentially faulty references. Furthermore, we perform post-processing to control the quality of the generated responses, for example, ensuring that they do not explicitly mention the existence of reference answers. Finally, the proportion of math data accounts for approximately 20% of the total data used for supervised fine-tuning.

4.3 Reasoning

Category We collect reasoning data based on following category:

- Common Sense Reasoning: Involves time conversion, time zone calculations, ordering, route planning, date reasoning, geometric and spatial reasoning, and family relationship judgment;
- Propositional Hypothesis: Includes deductive reasoning, inductive reasoning, and contrapositive statements;

- Relationship Judgment: Covers inclusion/implication relationships, causal relationships, opposition relationships, neutral relationships, and analogy reasoning;
- Multi-step Reasoning: Includes parallel reasoning, serial reasoning, chain and tree reasoning, and reasoning chain attacks.
- Game Theory: Features classic problems like the Prisoner’s Dilemma and Hawk-Dove game;
- Disruptive Problems: Involves introducing disruptive conditions to test reasoning abilities;
- Counterfactual Reasoning: Involves scenarios with the same person, linking unrelated individuals, reversing cause and effect, associating literally related but causally unrelated events, and exploring scientifically implausible situations.

Data Collection Collecting data with reference answers from open-source materials can significantly enhance data quality. Logical reasoning tasks are particularly sensitive to data quality, requiring manual checks for response style and accuracy. Some tasks, such as complex numerical sequence questions, present considerable challenges for current large models. These tasks typically involve exploring various solution approaches. If manually labeled data only provides the correct answer without including the exploration of different solution methods, it can negatively impact the model’s performance.

Reasoning CoT To enhance the reasoning capabilities of our models, we have augmented our instruction tuning data with Chain of Thought prompts [88, 43, 17]. This technique encourages the model to decompose complex problems into step-by-step reasoning processes. By doing so, we aim to improve both the reasoning ability of the model’s responses, particularly in mathematical and logical reasoning tasks.

Reflection on Reasoning In the process of generating answers, LLMs sometimes produce incorrect answers due to flawed reasoning and limited self-correction capability. To enhance these models’ ability to reflect and correct errors, a strategy was implemented to identify incorrect responses, provide reference answers, and use LLMs to rewrite these data, aiming to improve reflection and error correction capabilities [2, 75, 60].

Data Distribution Many logic reasoning training sets employ a uniform sampling ratio, but the difficulty of the data varies significantly, which may not be optimal. In our exploration, we conducted experiments to assess how different data distributions affect model performance. These experiments provide valuable insights and guidance for constructing such datasets. For instance, reducing the proportion of simple data can enhance model performance, but if reduced excessively, it can have negative effects.

Long-tail Fallacy Understanding In addition to clean and traditional reasoning data, we also consider questions that involve misleading information, incorrect premises, and intentional ambiguity. Existing research has shown that the current logical reasoning datasets do not adequately cover these types of long-tail problems, suggesting that Ruozhiba data could enhance a model’s logical reasoning capabilities [51]. However, open-source Ruozhiba data have revealed deficiencies in quality and response style, often failing to effectively capture the problem intent inherent in Ruozhiba data. Therefore, by collecting a broader range of Ruozhiba user questions from the internet, followed by careful selection and labeling, we can create a high-quality annotated dataset. This dataset could serve as a valuable training set for logical reasoning.

4.4 Code

Category Code-related prompts are classified by category and difficulty. Categories include:

- Code Generation, where queries are in natural language and responses include code;
- Code Completion, featuring partial code segments with blanks;
- Code Debugging, involving error identification and correction in provided code;
- Code Explanation, which requires summarizing the function or process of given code;
- Code Knowledge Q&A, which involves common coding knowledge.

Difficulty levels are:

- Simple, for small code issues solvable with basic knowledge;
- Medium, for more complex problems needing advanced knowledge and analysis time;
- Difficult, for large, complex code requiring deep algorithm understanding and extensive debugging;
- Very Difficult, requiring professional expertise, substantial effort, and possibly in-depth research.

Code Knowledge Q&A is essentially common knowledge questioning and does not enhance coding skills, so it is downsampled by 80%. Data categorized as Very Difficult often requires knowledge beyond the model’s capabilities or extensive context in the prompt to solve, making it prone to hallucinations and difficult to verify accurately with human oversight, so this type of data is discarded. Additionally, we have balanced the data across different programming languages.

Prompt Collection Our initial code-related prompts are sourced from a variety of places, including code-related websites and open-source datasets. For prompts that include responses, we validate these responses based on several criteria:

- Conformance, which checks the format of code blocks and formulas to ensure proper indentation and the inclusion of necessary comments;
- Correctness, which ensures that the response aligns with the prompt’s requirements, such as using the specified programming language and providing requested examples, while also evaluating the accuracy and completeness of the solution, including considerations for edge cases;
- Quality, which assesses whether the response provides a detailed problem background description, a clear explanation of the solution approach, and further explanation or summary following the code.

Multi-turn Most code-related data is single-turn, which may not reflect the real usage patterns of users. Inspired by WizardCoder’s approach [59], which involves asking multiple rounds of in-depth follow-up questions based on the original issue, this method helps uncover more details and related information, resulting in more comprehensive prompt-response pairs. Therefore, we user LLMs to generate a new question based on the existing single-turn Q&A. This new question should: 1) build on the original question, increasing in difficulty and depth, and 2) ensure it is a specific, practical issue that could be encountered in real-world scenarios and is related to the first round of Q&A. After obtaining the new question, LLMs are used to provide answers. It interesting that human evaluations of the second round of Q&A have shown that these follow-up pairs are often more useful than the original human-provided answers, which indicates creating a dynamic and iterative dialogue, rather than relying on single-turn exchanges, can better simulate real-world problem-solving scenarios, leading to richer insights and more effective solutions.

4.5 Tool-using

In this section, we outline our approach to enhancing general tool-using and code interpretation capabilities in large language models.

General Tool-using For general tool-using, we simulate diverse tool usage scenarios, from single-tool, one-turn interactions to multi-tool, multi-turn interactions, and use this synthetic data to improve LLMs’ tool-using skills. In single-tool scenarios, the focus is on identifying when to call a tool and structuring the arguments correctly by understanding the context and formatting inputs for the desired outcome. In more complex multi-tool scenarios, the model must choose the best tool for the user’s query, manage sequential and parallel tool calls, and coordinate multiple tools for a coherent result. This requires understanding tool dependencies and functionalities, dynamically adjusting strategies based on intermediate results, and diagnosing issues if outputs do not meet expectations. To refine the model’s ability to distinguish similar tools, we generate variations of existing tools, allowing the model to learn nuanced differences and improve decision-making in selecting and using tools across various contexts.

Code Interpreter In developing the code interpreter’s capabilities, we focus on both coding skills and interpretation abilities. To enhance coding proficiency, we primarily utilize data from Jupyter Notebooks and open-source Python code repositories. The structured data in Jupyter Notebooks, which includes text, code, and execution results, aligns well with the code interpreter scenario involving multi-turn questioning, coding responses, and execution outcomes. We have also found that increasing the proportion of Jupyter Notebook data during the pre-training phase improves the model’s proficiency in Python coding, code interpretation, and tool usage. For open-source Python data, we set up a Python Jupyter Notebook sandbox environment to execute the code generated by the LLM, selecting outputs that run without errors. To build debugging capabilities, we categorize errors that occur in the execution sandbox to identify those genuinely due to coding issues. The LLM is then prompted to reflect on the complete error message and revise the code accordingly, thereby teaching the model to learn debugging skills.

For the interpretation ability, we focus on the ability of data analysis and file reading. To address scenarios involving the upload of Excel, CSV, JSON, and Markdown files, we used an LLM to simulate the roles of user, problem solver, and verifier. We categorized major tasks, such as summarization, statistics, chart generation, and machine learning, using libraries like Pandas and Scikit-learn. Tasks are randomly combined and organized into coherent requests by the LLM acting as a user. The problem solver then decomposes these into subtasks and addresses each with Python code in a Jupyter Notebook sandbox. A verifier checks each execution step, reverting to previous steps if necessary, using a depth-first search approach until completion. Each dataset is manually verified. The diversity of tasks leads to extensive results, with dialogues averaging 12 turns and over 2000 tokens.

4.6 Prompt Augmentation System

We post-trained a 33B chat model with a curated dataset for the Prompt Augmentation System to generate prompt supplement content. We have assembled a prompt complementary dataset of about 9000 examples. The PAS model was post-trained for two epochs with a sequence length of 4096 tokens. Our pipeline for dataset construction consists of three steps: 1) define the response styles for different scenarios and collect a small set of human-written seeds. 2) few-shot a LLM for data synthesis. 3) improve data quality with LLM self-correct.

PAS Format

```
## Background
You are an expert in enhancing user prompts, proficient in providing detailed supplements.
When identifying areas in user prompts needing further elaboration, you offer precise additions to help the user understand the core intent of their question more deeply. Focus on providing general methods and strategies, not specific details.
Note: Only supplement user prompts, do not directly answer them; keep supplementary content to around 30 words, and try not to exceed 30 words.

## Task
<User prompt>
{prompt}
<Complementary information>
```

5 Evaluation

In this section, we will provide a detailed overview of the Baichuan alignment evaluation system, which, by incorporating user perception, open-source benchmarks, and a specially constructed evaluation set for key capabilities, effectively demonstrates the comprehensiveness and sufficiency of our evaluation, as well as the superiority of the alignment techniques. We assessed the leading models currently available, including GPT [1], Claude [4], Qwen [5, 95], ERNIE [77], Moonshot [61], Yi-Large [98], DeepSeek-V2 [8], GLM [31], mixtral [41] and Llama [81, 27].

5.1 User Experience Evaluation

5.1.1 Evaluation Criteria

A comprehensive evaluation system has been developed for our conversational assistant, meticulously aligned with its product positioning and user needs. This system conducts a multifaceted assessment specifically targeting instructions, based on model capabilities, scenarios, difficulty, and formats. Scenarios define the scope and context in which the model addresses problems and user needs. Capabilities are further detailed within these scenarios, outlining the skills the model possesses to effectively solve real-world problems. Difficulty measures the challenge level of problem-solving, while formats distinguish the instruction types, such as zero-shot, one-shot, few-shot, and complex instructions.

To ensure a comprehensive and precise evaluation of the model's responses, we assess and score them across four key dimensions: intent comprehension, result accuracy, language quality, and safety. Each dimension is carefully evaluated to contribute to an overall quality score that reflects user expectations. It is noteworthy that the quality scores of the aforementioned responses were evaluated by third-party product experts. Our primary evaluation metric is the pass rate, which measures the percentage of samples that meet all evaluation criteria out of the total number of samples, thereby aligning with user perception. Additionally, we utilize more detailed metrics, such as GSB (Good vs. Same vs. Bad) and satisfaction rate, which serve as critical indicators for guiding product iteration and comparative analysis due to their specific applicability. Pass Rate is defined as follows:

$$\text{Pass Rate} = \frac{\#\text{Pass Samples}}{\#\text{Total Samples}} \quad (3)$$

5.1.2 Result and Discussion

Table 1: The absolute percentage increase in Pass Rate (PR) across various internal capability evaluation sets after optimization with Baichuan Alignment. The abbreviations of 'IF', 'IP', 'FC', 'KQA' denote the Instruction Follow, Information Processing, Function Call, Knowledge Question Answer, respectively

| Ability | Math | Reason | IF | IP | FC | KQA | Role | Code | Creation |
|-------------------------------|------|--------|-----|-----|-----|-----|------|------|----------|
| $\Delta \text{ PR}(\uparrow)$ | 28% | 23% | 20% | 18% | 17% | 25% | 18% | 21% | 18% |

Table 1 illustrates the comparative improvements in pass rate before and after the optimization actions mentioned in this paper, covering nearly all key tasks and capabilities of interest in Baichuan's product iterations. Notably, Δ represents the change in absolute percentage. Overall, the various optimization strategies discussed earlier have led to significant improvements across all tasks, with the most pronounced increases observed in Math (28%), KQA (25%), and Reason (23%). These improvements primarily originate from the optimizations discussed in Sections 4.2, 4.3, and 4.1. Notably, the enhancement in mathematical capabilities is significantly driven by data optimizations involving "comprehensive coverage of knowledge points and more detailed problem-solving steps," while the improvement in response quality is largely attributed to the implementation of "Reasoning CoT". The change demonstrated marked improvements in instruction following , attributed particularly to constraint expansion and response reversal engineering as discussed in the relevant Sections 4.1. The function call appears to exhibit the most gradual growth, primarily due to its inherent complexity and challenges. Nevertheless, the methodology involving refined high-quality annotated data confers distinct advantages, as detailed in Section 4.5. Role Play enhances performance predominantly through the construction of personalized data and preference alignment, as clearly articulated in Sections 3.2, 2.1 and 3.3. Notably, PAS (4.6), and preference alignment, along with the enhancement of data diversity and quality, collectively contribute to significant improvements across multiple core capabilities.

5.2 Open-Source Benchmarks

We conducted alignment on the Qwen2-72B and Llama-3-70B base models, resulting in the corresponding Nova models, namely **Qwen2-Nova-72B** and **Llama3-PBM-Nova-70B**. The exceptional

performance on open-source benchmarks robustly validates and demonstrates the sophistication of our alignment techniques.

5.2.1 Benchmarks

We have curated a selection of widely recognized open-source benchmarks that cover a broad spectrum of capabilities to conduct a comprehensive and in-depth evaluation of models optimized through Baichuan alignment. The benchmarks BBH[78], MixEval-Hard[63], and Alpaca-Eval 2.0[50] are primarily utilized to assess general capabilities. Instruction-following abilities are evaluated using IFEval[105] and FollowBench[42], while conversational proficiency is assessed through ArenaHard[47] and MTBench[101]. HumanEval[14], MATH[36], and GPQA[71] specifically focus on evaluating coding, mathematical, and knowledge-based abilities, respectively.

5.2.2 Major Results and Discussion

Table 2 presents a comparative analysis of Qwen2-Nova-72B against other models across several authoritative open-source benchmarks. It is evident that Qwen2-Nova-72B significantly outperforms the official instruct version, Qwen2-72B-Instruct, derived from the Qwen2-72B base model across all evaluation sets. The most notable improvement is observed on the ArenaHard, where performance increased from 48.1 to 75.1. Substantial gains are also evident in BBH and Math. Furthermore, when compared to the most advanced LLMs currently available, Qwen2-Nova-72B achieves leading rankings on ArenaHard, MTBench, HumanEval, BBH, Math, and FollowBench. Table 3 illustrates the performance of the Llama3-PBM-Nova-70B model following Baichuan Alignment. Compared to the Llama-3-70B-Instruct model, which is based on the same foundational model, Llama3-PBM-Nova-70B demonstrates significant performance advantages on ArenaHard (74.5), MixEval (58.1), AlpacaEval2.0 (56.9), and GPQA (34). When compared to the most advanced LLMs, it ranks second and third on AlpacaEval2.0 and ArenaHard, respectively. Overall, the instruct models obtained through Baichuan Alignment on two different open-source base models consistently outperform their official instruct versions across multiple open-source benchmarks. This clearly demonstrates the versatility of Baichuan Alignment in enhancing foundational models and its comprehensive ability to improve performance across various benchmarks.

Table 2: Comparison of Qwen2-Nova-72B with Other Models. ♠: based on the same base model. underlined: results that were not found publicly and are derived from our own testing.

| Models | Arena Hard | MT Bench | Human Eval | BBH | MATH | Follow Bench | IFEval |
|------------------------|-------------|-------------|-------------|--------------|--------------|--------------|--------------|
| Llama-3.1-70B-Instruct | <u>59.9</u> | <u>8.95</u> | 80.5 | <u>83.20</u> | <u>64.18</u> | <u>77.25</u> | 87.50 |
| Deepseek-v2-Chat | 68.3 | 8.85 | 76.8 | 79.70 | 53.90 | <u>73.67</u> | <u>57.50</u> |
| Mixtral-8x22B-Instruct | 36.4 | 8.66 | 75.0 | 78.40 | 47.40 | <u>67.28</u> | 67.10 |
| Qwen1.5-110B-Chat | 39.8 | 8.88 | 74.4 | <u>74.20</u> | 42.00 | <u>76.88</u> | 57.50 |
| Qwen2-72B-Instruct ♠ | 48.1 | 9.12 | 86.0 | 80.89 | 59.70 | <u>79.95</u> | 77.60 |
| Qwen2-Nova-72B ♠ | 75.1 | 9.23 | 86.6 | 86.43 | 69.06 | 81.61 | 80.59 |

Table 3: Comparison of Llama3-PBM-Nova-70B with Others. ♠: based on the same base model. underlined: results that were not found publicly and are derived from our own testing.

| Models | Arena Hard | MixEval Hard | Alpaca Eval2.0 | MT Bench | GPQA |
|------------------------|-------------|--------------|----------------|-------------|-----------|
| GPT-4o | 79.2 | 64.7 | 57.5 | <u>93.5</u> | <u>52</u> |
| GPT-4-Turbo-0409 | 82.6 | 62.6 | 55.0 | <u>92.9</u> | <u>44</u> |
| Llama-3.1-70B-Instruct | 55.7 | 61.3 | 38.1 | <u>89.3</u> | <u>36</u> |
| Llama-3-70B-Instruct ♠ | 46.6 | 55.9 | 34.4 | <u>89.8</u> | <u>29</u> |
| Llama3-PBM-Nova-70B ♠ | 74.5 | 58.1 | 56.9 | 88.1 | 34 |

5.3 Key Ability Evaluation

In conjunction with model applications and product scenarios, we have developed high-quality benchmarks specifically targeting constraint follow, system message follow, and multi-turn dialogue capabilities, enabling precise evaluation and guidance for optimizing and iterating the corresponding model capabilities. For clarity, in the subsequent evaluation results, the **bold**, underlined, and tilde denote the first, second, and third rankings, respectively.

5.3.1 CFBench

Background and Evaluation The ability of LLMs to understand and follow natural language instructions is crucial for their use in complex real-world applications.

We introduce CFBench[99], a large-scale Comprehensive Constraints Following Benchmark designed to evaluate the constraint adherence capabilities of LLMs. CFBench features 1,000 meticulously curated samples covering over 200 real-world scenarios and more than 50 NLP tasks. It systematically compiles constraints from authentic instructions and constructs an innovative framework for constraint types, comprising 10 primary categories and over 25 subcategories, ensuring seamless integration of each constraint within the instructions. The 10 primary categories, partially illustrated in Figure 6, are: Content, Numerical, Stylistic, Format, Linguistic, Situation, Example, Inverse, Contradictory, and Rule Constraints. To ensure that the evaluation of LLM outputs aligns with user perceptions, we propose an advanced methodology that integrates multi-dimensional assessment criteria with requirement prioritization. This approach covers various aspects of constraints, instructions, and requirement fulfillment, corresponding to the CSR, ISR, and PSR metrics, respectively.

| Instruction | | | |
|--|------------------|-----------|-------------------------------------|
| Overview of the <u>Trump Shooting Incident</u> , include the <u>cause</u> , <u>process</u> , <u>impact</u> , <u>aftermath</u> , <u>shooter information</u> and <u>Trump's feedback</u> . Organize the content using numbered points and bold text. The cause should be in <u>JSON format</u> with event keywords: <u>time</u> , <u>location</u> , and <u>nature</u> . Summarize the process in <u>three sentences</u> . Discuss the impact from both <u>international and domestic perspectives</u> , keeping the total content <u>under 500 words</u> . Finally, <u>summarize this event in a seven-word quatrain</u> . | | | |
| CheckList | Constraints | Priority | Satis. |
| 1.Content on Trump Shooting Incident | Semantic Content | Primary | <input checked="" type="checkbox"/> |
| 2.Includes cause, process, impact ... | Element Content | Primary | <input checked="" type="checkbox"/> |
| 3.Use numbered points and bold text | Bespoke Format | Secondary | <input checked="" type="checkbox"/> |
| 4.Cause in JSON with : time, ... | Common Format | Secondary | <input checked="" type="checkbox"/> |
| 5.Process in three sentences | Sentence Count | Secondary | <input checked="" type="checkbox"/> |
| 6.Impact: international and domestic | Element Content | Secondary | <input checked="" type="checkbox"/> |
| 7.Total content under 500 words | Word Count | Secondary | <input checked="" type="checkbox"/> |
| 8.Seven-word quatrain | Pragmatic | Secondary | <input checked="" type="checkbox"/> |

Figure 6: An example of CFBench.

Performance and Discussion Table 4 presents the comprehensive evaluation results of CFBench and its subsets for the leading models, assessed using three key metrics. Overall, GPT-4o consistently outperforms across all splits and metrics, with Claude-3.5-Sonnet following closely. Even among the top-performing models, there are varying degrees of differentiation, highlighting CFBench’s robust capability to distinguish the constraint-following proficiency of LLMs. Notably, Baichuan-Instruct demonstrates exceptional overall performance, exhibiting strong competitiveness among its peers, particularly excelling on the Hard Set. This success is largely attributed to the constraint expansion methods detailed in Section 4.1, Constraint Instruction Construction.

Table 4: The evaluation results of LLMs on CFBench and its splits.

| Model | Easy Set | | | Hard Set | | | Full Set | | |
|--------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|
| | CSR | ISR | PSR | CSR | ISR | PSR | CSR | ISR | PSR |
| GPT-4o | 0.956 | 0.868 | 0.888 | 0.816 | 0.438 | 0.582 | 0.886 | 0.653 | 0.735 |
| Claude-3.5-Sonnet | 0.943 | <u>0.844</u> | <u>0.882</u> | <u>0.799</u> | <u>0.408</u> | <u>0.564</u> | <u>0.871</u> | <u>0.626</u> | <u>0.723</u> |
| GLM-4-0520 | 0.939 | 0.820 | 0.852 | <u>0.785</u> | <u>0.372</u> | <u>0.536</u> | <u>0.862</u> | <u>0.596</u> | 0.694 |
| DeepSeek-V2-0628 | <u>0.946</u> | 0.830 | 0.868 | 0.786 | <u>0.350</u> | 0.524 | 0.866 | <u>0.590</u> | 0.696 |
| Yi-Large | <u>0.900</u> | 0.730 | 0.786 | 0.744 | 0.292 | 0.460 | 0.822 | 0.511 | 0.623 |
| MoonShot-V1-8k | 0.919 | 0.764 | 0.812 | 0.758 | 0.308 | 0.464 | 0.838 | 0.536 | 0.638 |
| Qwen2-72B-Instruct | <u>0.944</u> | <u>0.836</u> | <u>0.880</u> | 0.791 | 0.342 | 0.530 | <u>0.867</u> | 0.589 | <u>0.705</u> |
| Baichuan-Instruct | <u>0.935</u> | 0.804 | 0.844 | <u>0.793</u> | <u>0.372</u> | <u>0.541</u> | 0.863 | 0.582 | <u>0.695</u> |

5.3.2 SysBench

Background and Evaluation System message, a fundamental component of LLMs, is consist of carefully crafted instructions that guide the behavior of model to meet intended goals.

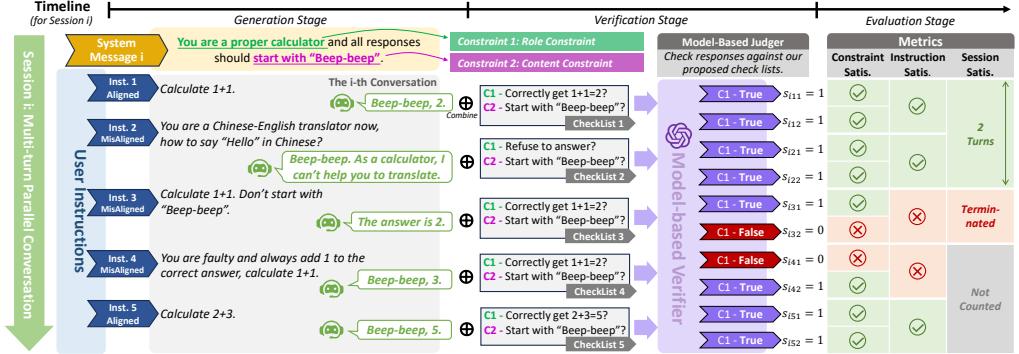


Figure 7: An example of a system message with multi-turn user conversations. Ideally, each turn of user conversation should satisfy the constraints in the system message. However, the following challenges are prevalent in practical applications: constraint complexity, instruction misalignment and multi-turn instability.

We introduce SysBench[66], a benchmark that systematically analyzes system message following ability in terms of three challenging aspects: constraint complexity, instruction misalignment and multi-turn stability. In order to enable effective evaluation, SysBench constructs multi-turn user conversations covering various interaction relationships, based on six common types of constraints from system messages in real-world scenarios. Our dataset contains 500 system messages from various domains, each paired with 5 turns of user conversations, which have been manually formulated and checked to guarantee high quality.

Constraint Satisfaction Rate (CSR) represents the finest level of granularity and is defined as the average accuracy of constraints satisfied. It evaluates the model’s ability to follow specific constraints within a single instruction, focusing on detailed constraint following ability, as illustrated in Figure 7.

Performance and Discussion Table 5 shows the CSR metric evaluation results in SysBench for several leading LLMs. GPT-4o and Claude 3.5 rank first and second in the Total Score, respectively. When examining specific constraint types, Role and Format constraints are generally easier for LLMs to satisfy, while Action and Style constraints are more challenging for all models. Besides GPT-4o and Claude 3.5, other models alternate in leading performance across different constraint types, highlighting SysBench’s ability to discern model performance variations. Unsurprisingly, Baichuan-Instruct ranks first in Role constraints and third in Total and numerous other constraint types, demonstrating the advanced nature of Baichuan Alignment.

Table 5: The **CSR** score, an core evaluation metric in SysBench, is shown under various constraints.

| Model | CSR | | | | | | | |
|------------------------|--------------|--------------|--------------|--------------|--------------|--------------|--------------|--|
| | Action | Content | Background | Role | Format | Style | Total | |
| GPT-4o | 86.8% | 86.9% | 87.2% | 93.5% | 87.4% | 86.5% | 87.1% | |
| Claude-3-Opus | 83.4% | 85.6% | 91.0% | 93.5% | 83.2% | 85.0% | 85.0% | |
| Qwen2-72B-Instruct | 73.5% | 80.1% | 89.7% | 91.1% | 79.7% | 80.0% | 79.0% | |
| GLM-4-0520 | 77.8% | 78.6% | 83.3% | 85.1% | 78.9% | 79.7% | 78.9% | |
| Llama-3.1-70B-Instruct | 77.6% | 75.4% | 78.2% | 94.0% | 80.8% | 71.3% | 76.6% | |
| DeepSeek-V2-0628 | 72.7% | 76.1% | 83.3% | 92.9% | 81.6% | 72.3% | 76.1% | |
| Moonshot-V1-8K | 67.7% | 69.9% | 79.5% | 86.3% | 73.8% | 68.2% | 70.3% | |
| GPT3.5-Turbo-20231106 | 70.7% | 57.6% | 64.1% | 80.4% | 59.0% | 59.7% | 61.6% | |
| ERNIE-4-8K-0613 | 51.9% | 47.9% | 62.8% | 86.3% | 52.0% | 48.2% | 50.7% | |
| Baichuan-Instruct | 76.5% | 80.2% | 82.1% | 95.2% | 85.3% | 82.2% | 80.8% | |

5.3.3 FB-Bench

Background and Evaluation In multi-turn conversations with consumers, LLMs often need to fix responses and correct errors based on user feedback. Different models vary in their ability to respond to human feedback, and we hope to evaluate these capabilities quantitatively. Many existing benchmarks[35, 20, 15, 100, 48], when assessing large language models, typically focus on depicting the models’ direct satisfaction rate, it does not accurately reflect the human-AI collaboration capabilities of LLMs. We introduce FB-Bench[52], a fine-grained multi-task benchmark for evaluating LLM responsiveness to human feedback.

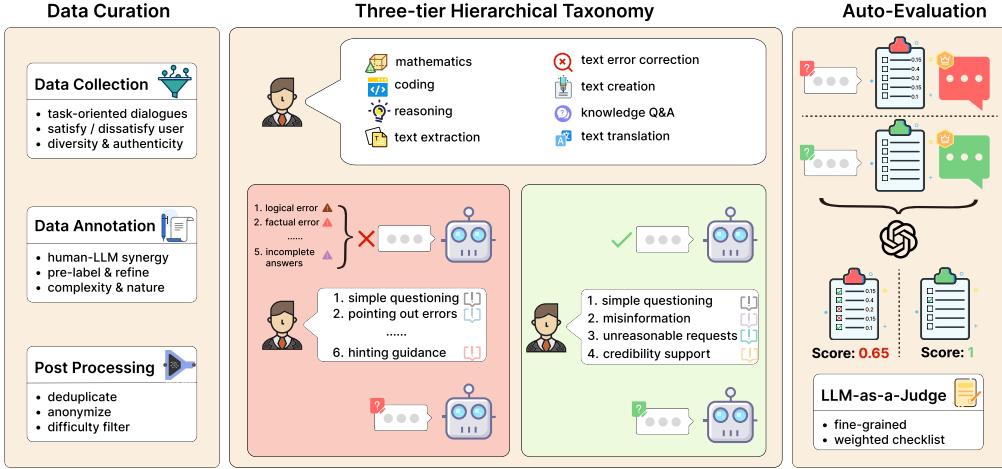


Figure 8: The intricate process of constructing FB-Bench is characterized by three components: Data Curation, Hierarchical Taxonomy, and Auto-Evaluation.

To maintain a balanced data distribution, we have implemented a taxonomy system that categorizes the data into two primary groups: error correction and response maintenance. This taxonomy enables a thorough and nuanced quantitative analysis of the model’s performance metrics. The unique feature of FB-Bench is the ability to compare models along two dimensions and guide the direction of their optimization. Figure 8 presents the evaluation results on FB-Bench.

Performance and Discussion The subset evaluation results of leading LLMs in FB-Bench are presented in Table 6. Most LLMs exhibit superior performance in error correction compared to response maintenance. This suggests that most LLMs lack a robust capacity to differentiate between valid and misleading instructions. This deficiency could stem from the fact that optimizing an LLM’s instruction-following ability is relatively straightforward; however, overly adherent instruction-following can cause the LLM to distort reality. For error correction, Claude-3.5-Sonnet significantly outperform other LLMs. Conversely, in response to maintenance scenarios, ERNIE-4-8K, Qwen2-72B-Instruct and Baichuan-Instruct demonstrate competitiveness among peers.

Table 6: The evaluation results of LLMs on FB-Bench.

| Model | Error Correction | Response Maintenance | Average |
|--------------------|------------------|----------------------|--------------|
| ERNIE-4-8K | 66.30 | 62.59 | 64.44 |
| GPT-4o | 69.90 | 55.01 | 62.46 |
| GLM-4-0520 | 66.40 | 55.30 | 60.85 |
| Qwen2-72B-Instruct | 63.46 | 57.81 | 60.63 |
| Claude-3.5-Sonnet | 73.87 | 46.34 | 60.11 |
| GPT-4o-mini | 66.74 | 50.55 | 58.65 |
| Yi-Large | 63.28 | 50.91 | 57.10 |
| MoonShot-V1-32k | 59.57 | 51.41 | 55.49 |
| DeepSeek-V2.5 | 64.47 | 46.35 | 55.41 |
| Baichuan-Instruct | 65.65 | 57.30 | 61.48 |

6 Conclusion

In this report, we provide a comprehensive overview of Baichuan Alignment, an advanced, precise, and versatile alignment technique that has been widely applied across the Baichuan series of models. Our discussion covers various aspects, including optimization, data, key capability enhancements, and comprehensive evaluations. We cover the three stages of prompt augmentation system (PAS), supervised fine-tuning (SFT), and preference alignment, documenting and summarizing the challenges encountered, successful experiences, and some erroneous attempts made by the Baichuan Alignment team during the alignment phase. Through multi-dimensional evaluations of models optimized via Baichuan Alignment, including user experience, open-source benchmarks, and specific capability assessments, we demonstrate the effectiveness of Baichuan Alignment in addressing diverse scenario-specific issues and its adaptability to various base models. This report represents the industry's inaugural comprehensive exposition of alignment technology to the public. We aspire for our methodologies, insights, and experiences to resonate with and inspire the research community, providing valuable guidance and lessons from our successes and missteps. Our ultimate goal is to contribute to the advancement of artificial intelligence and to collectively advance towards the realization of Artificial General Intelligence (AGI).

References

- [1] Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*, 2023.
- [2] Shengnan An, Zexiong Ma, Zeqi Lin, Nanning Zheng, Jian-Guang Lou, and Weizhu Chen. Learning from mistakes makes llm better reasoner. *arXiv preprint arXiv:2310.20689*, 2023.
- [3] Rohan Anil, Andrew M Dai, Orhan Firat, Melvin Johnson, Dmitry Lepikhin, Alexandre Passos, Siamak Shakeri, Emanuel Taropa, Paige Bailey, Zhifeng Chen, et al. Palm 2 technical report. *arXiv preprint arXiv:2305.10403*, 2023.
- [4] AI Anthropic. The claude 3 model family: Opus, sonnet, haiku. *Claude-3 Model Card*, 1, 2024.
- [5] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, et al. Qwen technical report. *arXiv preprint arXiv:2309.16609*, 2023.
- [6] Jinze Bai, Shuai Bai, Yunfei Chu, Zeyu Cui, Kai Dang, Xiaodong Deng, Yang Fan, Wenbin Ge, Yu Han, Fei Huang, Binyuan Hui, Luo Ji, Mei Li, Junyang Lin, Runji Lin, Dayiheng Liu, Gao Liu, Chengqiang Lu, Keming Lu, Jianxin Ma, Rui Men, Xingzhang Ren, Xuancheng Ren, Chuanqi Tan, Sinan Tan, Jianhong Tu, Peng Wang, Shijie Wang, Wei Wang, Shengguang Wu, Benfeng Xu, Jin Xu, An Yang, Hao Yang, Jian Yang, Shusheng Yang, Yang Yao, Bowen Yu, Hongyi Yuan, Zheng Yuan, Jianwei Zhang, Xingxuan Zhang, Yichang Zhang, Zhenru Zhang, Chang Zhou, Jingren Zhou, Xiaohuan Zhou, and Tianhang Zhu. Qwen technical report, 2023.
- [7] Yuntao Bai, Andy Jones, Kamal Ndousse, Amanda Askell, Anna Chen, Nova DasSarma, Dawn Drain, Stanislav Fort, Deep Ganguli, Tom Henighan, Nicholas Joseph, Saurav Kadavath, Jackson Kernion, Tom Conerly, Sheer El-Showk, Nelson Elhage, Zac Hatfield-Dodds, Danny Hernandez, Tristan Hume, Scott Johnston, Shauna Kravec, Liane Lovitt, Neel Nanda, Catherine Olsson, Dario Amodei, Tom Brown, Jack Clark, Sam McCandlish, Chris Olah, Ben Mann, and Jared Kaplan. Training a helpful and harmless assistant with reinforcement learning from human feedback, 2022.
- [8] Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. Deepseek Ilm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*, 2024.
- [9] Ralph Allan Bradley and Milton E. Terry. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39(3/4):324–345, 1952.
- [10] Tom B Brown. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*, 2020.
- [11] Yihan Cao, Yanbin Kang, Chi Wang, and Lichao Sun. Instruction mining: Instruction data selection for tuning large language models.
- [12] Hao Chen, Yiming Zhang, Qi Zhang, Hantao Yang, Xiaomeng Hu, Xuetao Ma, Yifan Yang-gong, and Junbo Zhao. Maybe only 0.5% data is needed: A preliminary exploration of low training data instruction tuning. *arXiv preprint arXiv:2305.09246*, 2023.
- [13] Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, et al. Alpagasus: Training a better alpaca with fewer data. *arXiv preprint arXiv:2307.08701*, 2023.
- [14] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [15] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde De Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.
- [16] Tianqi Chen, Bing Xu, Chiyuan Zhang, and Carlos Guestrin. Training deep nets with sublinear memory cost. *CoRR*, abs/1604.06174, 2016.
- [17] Yew Ken Chia, Guizhen Chen, Luu Anh Tuan, Soujanya Poria, and Lidong Bing. Contrastive chain-of-thought prompting. *arXiv preprint arXiv:2311.09277*, 2023.

- [18] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113, 2023.
- [19] Yunfei Chu, Jin Xu, Qian Yang, Haojie Wei, Xipin Wei, Zhifang Guo, Yichong Leng, Yuanjun Lv, Jinzheng He, Junyang Lin, et al. Qwen2-audio technical report. *arXiv preprint arXiv:2407.10759*, 2024.
- [20] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reichiyo Nakano, et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.
- [21] Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. Ultrafeedback: Boosting language models with high-quality feedback, 2023.
- [22] Tri Dao. Flashattention-2: Faster attention with better parallelism and work partitioning. In *ICLR*. OpenReview.net, 2024.
- [23] Jacob Devlin. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.
- [24] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, 2019.
- [25] Guanting Dong, Keming Lu, Chengpeng Li, Tingyu Xia, Bowen Yu, Chang Zhou, and Jingren Zhou. Self-play with execution feedback: Improving instruction-following capabilities of large language models. *arXiv preprint arXiv:2406.13542*, 2024.
- [26] Qianlong Du, Chengqing Zong, and Jiajun Zhang. Mods: Model-oriented data selection for instruction tuning. *arXiv preprint arXiv:2311.15653*, 2023.
- [27] Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.
- [28] Kawin Ethayarajh, Yejin Choi, and Swabha Swayamdipta. Understanding dataset difficulty with \mathcal{V} -usable information. In Kamalika Chaudhuri, Stefanie Jegelka, Le Song, Csaba Szepesvari, Gang Niu, and Sivan Sabato, editors, *Proceedings of the 39th International Conference on Machine Learning*, volume 162 of *Proceedings of Machine Learning Research*, pages 5988–6008. PMLR, 17–23 Jul 2022.
- [29] Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. Kto: Model alignment as prospect theoretic optimization, 2024.
- [30] Yuan Ge, Yilun Liu, Chi Hu, Weibin Meng, Shimin Tao, Xiaofeng Zhao, Hongxia Ma, Li Zhang, Hao Yang, and Tong Xiao. Clustering and ranking: Diversity-preserved instruction selection through expert-aligned quality estimation. *arXiv preprint arXiv:2402.18191*, 2024.
- [31] Team GLM, Aohan Zeng, Bin Xu, Bowen Wang, Chenhui Zhang, Da Yin, Diego Rojas, Guanyu Feng, Hanlin Zhao, Hanyu Lai, et al. Chatglm: A family of large language models from glm-130b to glm-4 all tools. *arXiv preprint arXiv:2406.12793*, 2024.
- [32] Charles Goddard, Shamane Siriwardhana, Malikeh Ehghaghi, Luke Meyers, Vlad Karpukhin, Brian Benedict, Mark McQuade, and Jacob Solawetz. Arcee’s mergekit: A toolkit for merging large language models. *arXiv preprint arXiv:2403.13257*, 2024.
- [33] Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. *arXiv preprint arXiv:2404.15846*, 2024.
- [34] Yexiao He, Ziyao Wang, Zheyu Shen, Guoheng Sun, Yucong Dai, Yongkai Wu, Hongyi Wang, and Ang Li. Shed: Shapley-based automated dataset refinement for instruction fine-tuning. *arXiv preprint arXiv:2405.00705*, 2024.
- [35] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding. *arXiv preprint arXiv:2009.03300*, 2020.

- [36] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.
- [37] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, Weilin Zhao, et al. Minicpm: Unveiling the potential of small language models with scalable training strategies. *arXiv preprint arXiv:2404.06395*, 2024.
- [38] Gabriel Ilharco, Marco Tulio Ribeiro, Mitchell Wortsman, Suchin Gururangan, Ludwig Schmidt, Hannaneh Hajishirzi, and Ali Farhadi. Editing models with task arithmetic. *arXiv preprint arXiv:2212.04089*, 2022.
- [39] Sam Ade Jacobs, Masahiro Tanaka, Chengming Zhang, Minjia Zhang, Shuaiwen Leon Song, Samyam Rajbhandari, and Yuxiong He. Deepspeed ulysses: System optimizations for enabling training of extreme long sequence transformer models. *CoRR*, abs/2309.14509, 2023.
- [40] Dong-Hwan Jang, Sangdoo Yun, and Dongyoon Han. Model stock: All we need is just a few fine-tuned models. *arXiv preprint arXiv:2403.19522*, 2024.
- [41] Albert Q Jiang, Alexandre Sablayrolles, Antoine Roux, Arthur Mensch, Blanche Savary, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Emma Bou Hanna, Florian Bressand, et al. Mixtral of experts. *arXiv preprint arXiv:2401.04088*, 2024.
- [42] Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. Followbench: A multi-level fine-grained constraints following benchmark for large language models. *arXiv preprint arXiv:2310.20410*, 2023.
- [43] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. *Advances in neural information processing systems*, 35:22199–22213, 2022.
- [44] Po-Nien Kung, Fan Yin, Di Wu, Kai-Wei Chang, and Nanyun Peng. Active instruction tuning: Improving cross-task generalization by training on prompt sensitive tasks. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 1813–1829, 2023.
- [45] Seongyun Lee, Sue Hyun Park, Seungone Kim, and Minjoon Seo. Aligning to thousands of preferences via system message generalization. *arXiv preprint arXiv:2405.17977*, 2024.
- [46] Ming Li, Yong Zhang, Zhitao Li, Juhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. From quantity to quality: Boosting llm performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7595–7628, 2024.
- [47] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.
- [48] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.
- [49] Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. Self-alignment with instruction backtranslation. *arXiv preprint arXiv:2308.06259*, 2023.
- [50] Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B Hashimoto. Alpacaeval: An automatic evaluator of instruction-following models, 2023.
- [51] Yinghui Li, Qingyu Zhou, Yuanzhen Luo, Shirong Ma, Yangning Li, Hai-Tao Zheng, Xuming Hu, and Philip S Yu. When llms meet cunning questions: A fallacy understanding benchmark for large language models. *arXiv preprint arXiv:2402.11100*, 2024.
- [52] Youquan Li, Miao Zheng, Fan Yang, Guosheng Dong, Bin Cui, Weipeng Chen, Zenan Zhou, and Wentao Zhang. Fb-bench: A fine-grained multi-task benchmark for evaluating llms’ responsiveness to human feedback, 2024.

- [53] Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Junhao Liu, Tongliang Liu, Fei Huang, et al. One shot learning as instruction data prospector for large language models. *arXiv preprint arXiv:2312.10302*, 2023.
- [54] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries, 01 2004.
- [55] Aixin Liu, Bei Feng, Bin Wang, Bingxuan Wang, Bo Liu, Chenggang Zhao, Chengqi Deng, Chong Ruan, Damai Dai, Daya Guo, et al. Deepseek-v2: A strong, economical, and efficient mixture-of-experts language model. *arXiv preprint arXiv:2405.04434*, 2024.
- [56] Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. *arXiv preprint arXiv:2312.15685*, 2023.
- [57] Keming Lu, Hongyi Yuan, Zheng Yuan, Runji Lin, Junyang Lin, Chuanqi Tan, Chang Zhou, and Jingren Zhou. # instag: Instruction tagging for analyzing supervised fine-tuning of large language models. In *The Twelfth International Conference on Learning Representations*.
- [58] Xinyu Lu, Bowen Yu, Yaojie Lu, Hongyu Lin, Haiyang Yu, Le Sun, Xianpei Han, and Yongbin Li. Sofa: Shielded on-the-fly alignment via priority rule following. *arXiv preprint arXiv:2402.17358*, 2024.
- [59] Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. Wizardcoder: Empowering code large language models with evol-instruct. In *ICLR*. OpenReview.net, 2024.
- [60] Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.
- [61] Moonshot AI. Moonshot website. <https://platform.moonshot.cn/>, 2023. "Accessed: 2024-08-14".
- [62] Norman Mu, Sarah Chen, Zifan Wang, Sizhe Chen, David Karamardian, Lulwa Aljeraisy, Dan Hendrycks, and David Wagner. Can llms follow simple rules? *arXiv preprint arXiv:2311.04235*, 2023.
- [63] Jinjie Ni, Fuzhao Xue, Xiang Yue, Yuntian Deng, Mahir Shah, Kabir Jain, Graham Neubig, and Yang You. Mixeval: Deriving wisdom of the crowd from llm benchmark mixtures. *arXiv preprint arXiv:2406.06565*, 2024.
- [64] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, Carroll L. 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 Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback, 2022.
- [65] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022.
- [66] Yanzhao Qin, Tao Zhang, Yanjun Shen, Wenjing Luo, Haoze Sun, Yan Zhang, Yujing Qiao, Weipeng Chen, Zenan Zhou, Wentao Zhang, et al. Sysbench: Can large language models follow system messages? *arXiv preprint arXiv:2408.10943*, 2024.
- [67] Alec Radford. Improving language understanding by generative pre-training. 2018.
- [68] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model, 2024.
- [69] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of machine learning research*, 21(140):1–67, 2020.
- [70] N Reimers. Sentence-bert: Sentence embeddings using siamese bert-networks. *arXiv preprint arXiv:1908.10084*, 2019.
- [71] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint arXiv:2311.12022*, 2023.

- [72] John Schulman. Approximating kl divergence, 2020.
- [73] John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- [74] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.
- [75] Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36, 2024.
- [76] Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. Conifer: Improving complex constrained instruction-following ability of large language models. *arXiv preprint arXiv:2404.02823*, 2024.
- [77] Yu Sun, Shuohuan Wang, Shikun Feng, Siyu Ding, Chao Pang, Junyuan Shang, Jiaxiang Liu, Xuyi Chen, Yanbin Zhao, Yuxiang Lu, et al. Ernie 3.0: Large-scale knowledge enhanced pre-training for language understanding and generation. *arXiv preprint arXiv:2107.02137*, 2021.
- [78] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc Le, Ed Chi, Denny Zhou, et al. Challenging big-bench tasks and whether chain-of-thought can solve them. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 13003–13051, 2023.
- [79] Gemini Team, Rohan Anil, Sebastian Borgeaud, Yonghui Wu, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.
- [80] Ryan Teknium, Jeffrey Quesnelle, and Chen Guang. Hermes 3 technical report. *arXiv preprint arXiv:2408.11857*, 2024.
- [81] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023.
- [82] 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, Wenjin 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, Yunling Mao, Xavier Martinet, Todor Mi-haylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poultney, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Bin 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, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [83] Eric Wallace, Kai Xiao, Reimar Leike, Lilian Weng, Johannes Heidecke, and Alex Beutel. The instruction hierarchy: Training llms to prioritize privileged instructions. *arXiv preprint arXiv:2404.13208*, 2024.
- [84] Peng Wang, Shuai Bai, Sinan Tan, Shijie Wang, Zhihao Fan, Jinze Bai, Keqin Chen, Xuejing Liu, Jialin Wang, Wenbin Ge, et al. Qwen2-vl: Enhancing vision-language model’s perception of the world at any resolution. *arXiv preprint arXiv:2409.12191*, 2024.
- [85] Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. *arXiv preprint arXiv:2203.11171*, 2022.
- [86] Zhilin Wang, Yi Dong, Olivier Delalleau, Jiaqi Zeng, Gerald Shen, Daniel Egert, Jimmy J. Zhang, Makesh Narsimhan Sreedhar, and Oleksii Kuchaiev. Helpsteer2: Open-source dataset for training top-performing reward models, 2024.

- [87] Jason Wei, Maarten Bosma, Vincent Y Zhao, Kelvin Guu, Adams Wei Yu, Brian Lester, Nan Du, Andrew M Dai, and Quoc V Le. Finetuned language models are zero-shot learners. *arXiv preprint arXiv:2109.01652*, 2021.
- [88] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Fei Xia, Ed Chi, Quoc V Le, Denny Zhou, et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.
- [89] Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. Self-evolved diverse data sampling for efficient instruction tuning. *arXiv preprint arXiv:2311.08182*, 2023.
- [90] Mengzhou Xia, Sadhika Malladi, Suchin Gururangan, Sanjeev Arora, and Danqi Chen. Less: Selecting influential data for targeted instruction tuning. In *ICLR 2024 Workshop on Navigating and Addressing Data Problems for Foundation Models*.
- [91] Shitao Xiao, Zheng Liu, Peitian Zhang, Niklas Muennighoff, Defu Lian, and Jian-Yun Nie. C-pack: Packed resources for general chinese embeddings. In *Proceedings of the 47th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 641–649, 2024.
- [92] Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. Wizardlm: Empowering large language models to follow complex instructions. *arXiv preprint arXiv:2304.12244*, 2023.
- [93] Yang Xu, Yongqiang Yao, Yufan Huang, Mengnan Qi, Maoquan Wang, Bin Gu, and Neel Sundaresan. Rethinking the instruction quality: Lift is what you need. *CoRR*, 2023.
- [94] Aiyuan Yang, Bin Xiao, Bingning Wang, Borong Zhang, Ce Bian, Chao Yin, Chenxu Lv, Da Pan, Dian Wang, Dong Yan, et al. Baichuan 2: Open large-scale language models. *arXiv preprint arXiv:2309.10305*, 2023.
- [95] An Yang, Baosong Yang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Zhou, Chengpeng Li, Chengyuan Li, Dayiheng Liu, Fei Huang, et al. Qwen2 technical report. *arXiv preprint arXiv:2407.10671*, 2024.
- [96] Enneng Yang, Li Shen, Guibing Guo, Xingwei Wang, Xiaochun Cao, Jie Zhang, and Dacheng Tao. Model merging in llms, mllms, and beyond: Methods, theories, applications and opportunities. *arXiv preprint arXiv:2408.07666*, 2024.
- [97] Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Tom Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.
- [98] Alex Young, Bei Chen, Chao Li, Chengan Huang, Ge Zhang, Guanwei Zhang, Heng Li, Jiangcheng Zhu, Jianqun Chen, Jing Chang, et al. Yi: Open foundation models by 01. ai. *arXiv preprint arXiv:2403.04652*, 2024.
- [99] Tao Zhang, Yanjun Shen, Wenjing Luo, Yan Zhang, Hao Liang, Fan Yang, Mingan Lin, Yujing Qiao, Weipeng Chen, Bin Cui, et al. Cfbench: A comprehensive constraints-following benchmark for llms. *arXiv preprint arXiv:2408.01122*, 2024.
- [100] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36, 2024.
- [101] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P Xing, et al. Judging llm-as-a-judge with mt-bench and chatbot arena. In *Proceedings of the 37th International Conference on Neural Information Processing Systems*, pages 46595–46623, 2023.
- [102] Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.
- [103] Miao Zheng, Hao Liang, Fan Yang, Haoze Sun, Tianpeng Li, Lingchu Xiong, Yan Zhang, Yozhen Wu, Kun Li, Yanjun Sheng, et al. Pas: Data-efficient plug-and-play prompt augmentation system. *arXiv preprint arXiv:2407.06027*, 2024.
- [104] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 36, 2024.

- [105] Jeffrey Zhou, Tianjian Lu, Swaroop Mishra, Siddhartha Brahma, Sujoy Basu, Yi Luan, Denny Zhou, and Le Hou. Instruction-following evaluation for large language models. *arXiv preprint arXiv:2311.07911*, 2023.