A Survey of Post-Training Scaling in Large Language Models

Hanyu Lai[1*†], **Xiao Liu**[1,2*], **Junjie Gao**[3*†], **Jiale Cheng**[1†], **Zehan Qi**[1†], **Yifan Xu**[1†], **Shuntian Yao**[1†], **Dan Zhang**[1†], **Jinhua Du**[1†], **Zhenyu Hou**[1,2], **Xin Lv**[2], **Minlie Huang**[1], **Yuxiao Dong**[1], **Jie Tang**[1‡]

[1] Tsinghua University [2] Zhipu AI [3] MBZUAI

Abstract

Large language models (LLMs) have achieved remarkable proficiency in understanding and generating human natural languages, mainly owing to the "scaling law" that optimizes relationships among language modeling loss, model parameters, and pre-trained tokens. However, with the exhaustion of high-quality internet corpora and increasing computational demands, the sustainability of pre-training scaling needs to be addressed. This paper presents a comprehensive survey of post-training scaling, an emergent paradigm aiming to relieve the limitations of traditional pre-training by focusing on the alignment phase, which traditionally accounts for a minor fraction of the total training computation. Our survey categorizes post-training scaling into three key methodologies: Supervised Fine-tuning (SFT), Reinforcement Learning from Feedback (RLxF), and Test-time Compute (TTC). We provide an in-depth analysis of the motivation behind post-training scaling, the scalable variants of these methodologies, and a comparative discussion against traditional approaches. By examining the latest advancements, identifying promising application scenarios, and highlighting unresolved issues, we seek a coherent understanding and map future research trajectories in the landscape of post-training scaling for LLMs.

1 Introduction

Large Language Models (LLMs) (Brown, 2020; Chowdhery et al., 2023; Hoffmann et al., 2022; Zhang et al., 2022; Zeng et al., 2022; Touvron et al., 2023; Le Scao et al., 2023) have demonstrated unprecedented capabilities to understand and generate human natural languages. They are self-supervisedly (Liu et al., 2021) pre-trained over trillion-scale internet corpus, covering a broad spectrum of potential contents of language (Gao et al., 2020; Yuan et al., 2021; Penedo et al., 2024), coding (Kocetkov et al., 2022; Xia et al., 2024), mathematics (Wang et al., 2023d), and other professional or scientific knowledge (Lo et al., 2019), to gain a firm grasp of commonsense, world knowledge (Xue et al., 2024), and even emergent abilities (Wei et al., 2022b) to reason like humans.

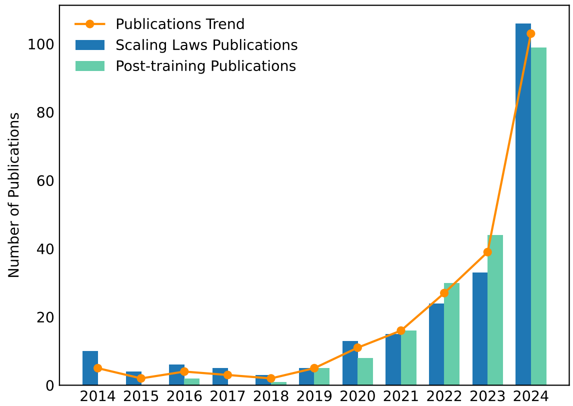

Figure 1: The number of publications on Scaling Laws and Post-training from arXiv and Google Scholar.

The success of LLMs heavily depends on "Scaling Law" (Brown, 2020; Hoffmann et al., 2022) in pre-training, which unveils the numerical relationships of the language modeling loss, model parameters, and pre-trained tokens. By fully exploiting the potential of data and parameters according to the scaling law, LLMs such as GPT-4 (OpenAI, 2024a) have significantly outperformed average humans in writing and language understanding, and have even matched the performance of undergraduate students on disciplinary examinations. As a result, the scaling law has become a critical foundation of contemporary LLMs.

However, as high-quality internet corpus becomes potentially exhausted (Villalobos et al.), the pre-training scaling is facing a substantial challenge. While massively synthesizing corpus might tackle the problem, its effectiveness after scaling remains unverified and questionable (Shumailov et al., 2024). Additionally, the extreme compu-

*LH, LX and GJ contributed equally.
†Work done while these authors interned at Zhipu AI.
‡Corresponding author.

Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 2771–2791
July 27 - August 1, 2025

tation required by pre-training has cast doubt on the marginal benefits and return on investment of pre-training scaling. Consequently, it becomes increasingly a consensus (Figure 1) that we need *scaling laws beyond pre-training* to achieve artificial general intelligence (AGI).

The advent of OpenAI o1 model (OpenAI, 2024b), together with some recent works (Snell et al., 2024; Yue et al., 2023; Zhang et al., 2024a), has been advocating another vital stream of scaling: **Post-training Scaling**. Instead of investing in the self-supervised pre-training phase, post-training scaling emphasizes the post-training phase (i.e., alignment), which conventionally only accounts for less than 1% of the whole LLM training computation. The probable improving aspects include Supervised Fine-tuning (SFT), Reinforcement Learning from "X" Feedback (RLxF), and Test-time Compute (TTC) (i.e., inference scaling).

In this work, we aim to present a comprehensive survey on methods for the new scaling laws: post-training scaling. We first provide an overview of the motivation for developing post-training scaling and then dive into the three summarized categories of methods above. While these categories are established, our surveying emphasis lies in their *scalable* variants that could illuminate the post-training scaling challenge. For comparison, we briefly introduce common post-training strategies that do not scale well enough in each category. To sum up, in this survey, we make the following contributions:

- We meticulously examine the latest advancements in post-training methodologies, thoroughly overview the fundamental concepts, training techniques, and critical frameworks, and aim to facilitate an in-depth understanding of these cutting-edge developments.
- We categorize post-training scaling laws into three distinct stages: Supervised Fine-tuning (SFT), Reinforcement Learning (RL), and Test-time Compute (TTC). We compare traditional and scalable approaches for each stage, highlighting their respective advantages and disadvantages.
- We identify several promising applications within the field and discuss unresolved issues, analyzing their limitations and boundaries. Our discussion extends to future directions for post-training scaling laws, mapping out potential trajectories for continued research and development.

This survey is structured as follows. Section 2 outlines the motivation behind post-training scaling for enhancing LLMs. Sections 3, 4, and 5 explore scalable methodologies within Supervised Fine-tuning, Reinforcement Learning, and Test-time Compute, comparing traditional and scalable methods. Section 6 examines the applications of these post-training techniques in areas such as mathematics, coding, and autonomous agents. Section 7 concludes with key insights from our comprehensive survey. The limitations section discusses unresolved issues and future research directions.

2 Motivation of Post-Training Scaling

The concept of post-training has a long-standing history (Moreau and Audiffren, 2017). Unlike self-supervised pre-training, which primarily learns language's statistical properties and fundamental semantics, post-training further enlightens the model through alignment and guidance techniques. The increasing research volume underscores the importance of post-training (Touvron et al., 2023; OpenAI, 2024a), noting its evolution from incremental training for alignment to fostering a learning process where models exhibit autonomous reasoning. The OpenAI o1 model (OpenAI, 2024b) suggests that a scaling law persists during the post-training phase, offering alternative strategies when existing data are insufficient for further training. Consequently, the automation and scaling of post-training processes are pivotal for advancing LLMs to the next level. Traditionally, post-training can be categorized into three main types:

- Supervised Fine-tuning: The LLM is trained to map input instructions to output labels.
- Reinforcement Learning: The LLM generates outputs based on input instructions, receives reward signals from the environment, and updates itself according to these rewards.
- Test-time Compute: The LLM utilizes computation and inference strategies to enhance performance across various scenarios.

Given the rapid iteration of post-training algorithms, we aim to classify them and review their scaling potential to facilitate further advancements. Below is an overview of our classification:

- Supervised Fine-tuning: SFT encompasses methods for generating instructions and responses. We classify these methods and analyze their potential for scaling with larger datasets or extended training.
- Reinforcement Learning: The reward signal is

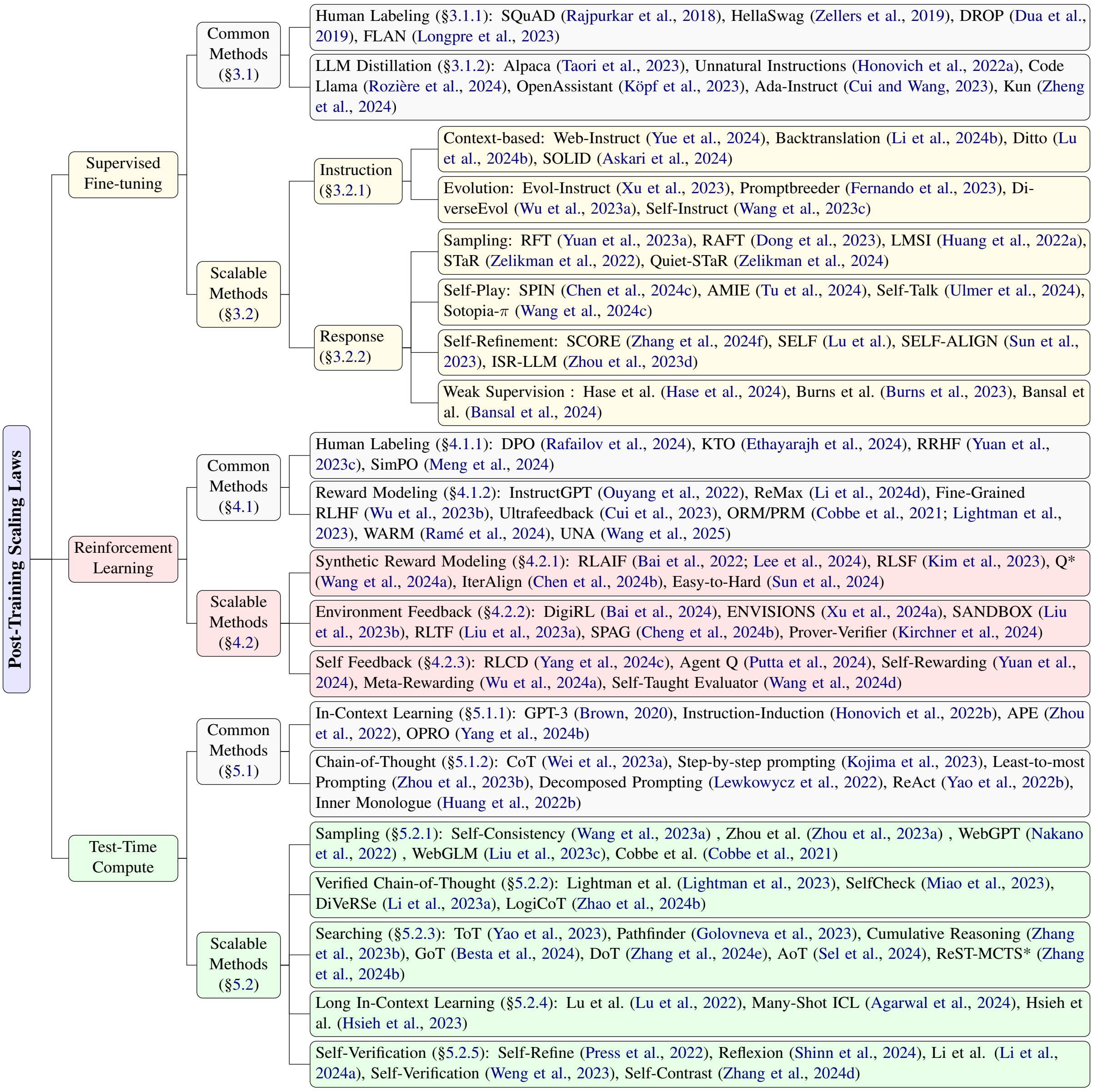

Figure 2: Taxonomy of Post-Training Scaling Laws.

an essential component of RL. We classify RL methods based on the source of the reward signal and evaluate the automation and scalability of these signals.

- Test-time Compute: We classify methods based on the target and approach of scaling computations. Additionally, we examine whether these methods can improve performance through increased computations.

We classify the existing post-training methods and present a tree diagram, as shown in Figure 2. The tree diagram summarizes the three core approaches to post-training: Supervised Fine-tuning, Reinforcement Learning, and Test-time Compute. Each approach is further subdivided into common and scalable methods.

By providing a structured classification and their methods (Figure 2), we aim to contribute to the discussion on the post-training algorithms and inspire further research and development in this area.

3 Scaling for Supervised Fine-tuning

Supervised Fine-tuning (SFT) is a training technique aimed at improving the performance of pre-trained language models on a specific task by training on a supervised high-quality labeled dataset (Sanh et al., 2022; Wei et al., 2022a). InstructGPT (Ouyang et al., 2022) and ChatGPT (OpenAI, 2022) leverage SFT after pre-training to improve performance on specific tasks

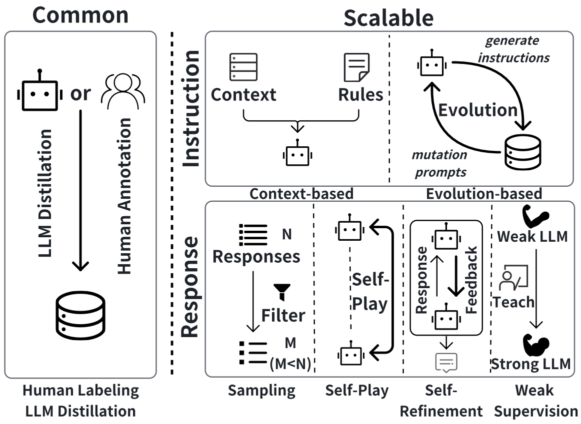

Figure 3: Classification of SFT Methods.

like answering questions and following users' instructions. Scaling for SFT involves constructing instructions and responses through various scalable training and data-constructing methods, like Web-Instruct (Yue et al., 2024), Evol-Instruct (Xu et al., 2023) and STaR (Zelikman et al., 2022). These methods can be categorized based on their data acquisition techniques (Figure 3). This section summarizes the common and scalable methods for SFT from the data acquisition perspective.

3.1 The Common Methods

Previous research on SFT mainly uses human labeling (Bach et al., 2022; Longpre et al., 2023) or LLMs like ChatGPT and GPT-4 (OpenAI, 2022, 2024a) with prompt engineering. **Human labeling** produces high-quality data but is costly and insufficient for scaling, while **LLM distillation** produces plentiful data but often lacks robust quality and diversity (Bender et al., 2021; Brown et al., 2020).

3.1.1 Human Labeling

Early approaches to SFT in NLP rely heavily on human labeling, exemplified by widely-used public datasets such as SQuAD (Rajpurkar et al., 2018), HellaSwag (Zellers et al., 2019), DROP (Dua et al., 2019) and FLAN (Longpre et al., 2023). They leverage human-labeled instruction data to enhance their ability to generate text aligned with human-provided instructions, improving response quality.

3.1.2 LLM Distillation

Researchers increasingly use LLMs, such as ChatGPT (OpenAI, 2022), to generate task-specific data efficiently and inexpensively, though ensuring data accuracy remains challenging due to the limitations of the LLMs. Alpaca (Taori et al., 2023) employs prompt engineering with `text-davinci-003` for creating instruction-response pairs from initial tasks. Unnatural Instructions (Honovich et al., 2022a) combines manual and automated processes for quality control in data generation. Code Llama (Rozière et al., 2024) and Openassistant (Köpf et al., 2023) use advanced prompting techniques with Llama-2 70B (Touvron et al., 2023) to generate program solutions. Fine-tuning existing models with high-quality data is another strategy. Ada-Instruct (Cui and Wang, 2023) fine-tunes with few shots for efficient instruction generators, while Kun (Zheng et al., 2024) uses a dual-model approach to label and refine data.

3.2 The Scalable Methods

In this section, we categorize the scaling techniques based on the data sources, distinguishing between **instruction** and **response** generation, and further detail their respective construction methodologies.

3.2.1 Instruction Generation

Context-based methods augment LLMs by integrating external knowledge into inputs, enhancing the instructions' diversity and authenticity. However, they often incur costs and complexities in knowledge collection. For instance, Web-Instruct (Yue et al., 2024) retrieves and contextualizes documents from web databases to create an instruction dataset. At the same time, Backtranslation (Li et al., 2024b) utilizes 502K unlabeled text segments from the ClueWeb corpus to iteratively fine-tune a seed model and create high-quality training examples. Ditto (Lu et al., 2024b) employs a self-alignment method to simulate dialogues based on 4,000 high-quality role knowledge entries, and SOLID (Askari et al., 2024) automates prompt generation for dialogue through multi-intent instructions, amassing extensive data.

Evolution-based techniques mimic natural processes to refine instructions iteratively, thereby boosting diversity and quality through mutation prompts or self-instruction mechanisms. Evol-Instruct (Xu et al., 2023) uses deep and breadth evolution strategies to generate instructions of varied difficulty levels automatically. Promptbreeder (Fernando et al., 2023) employs an evolutionary algorithm with mutation prompts to improve task prompts via a binary tournament genetic process. DiverseEvol (Wu et al., 2023a) enhances data diversity by using a K-Center sampling algorithm to select the most dissimilar data points iteratively. Moreover, Self-instruct (Wang et al., 2023c) allows a model to autonomously generate and validate new task instructions from a seed dataset, gradually building a comprehensive pool of tasks.

3.2.2 Response Generation

Sampling strategies generate multiple candidate responses to a given instruction, selecting the highest-quality output using distinct algorithms. RFT (Yuan et al., 2023a) leverages a rejection sampling algorithm to filter reasoning paths, ensuring high-quality dataset generation. RAFT (Dong et al., 2023) enhances model performance through reward-ranked fine-tuning - iteratively generating and selecting optimal responses. LMSI (Huang et al., 2022a) refines predictions using Chain-of-Thought prompts and a majority voting mechanism, optimizing model fine-tuning. STaR (Zelikman et al., 2022) and Quiet-STaR (Zelikman et al., 2024) further advance LLM reasoning by sampling rationales, with Quiet-STaR integrating parallel sampling for more diverse rationale generation.

Self-Play is a significant avenue in model improvement, involving the model iterating against itself to refine strategies. This approach, rooted in game theory and illustrated by early checkers research (Samuel, 1959), finds practical applications in various fields. SPIN (Chen et al., 2024c) implements self-play within LLMs to boost performance without labeled data. AMIE (Tu et al., 2024) uniquely applies self-play in the medical field, iterating internal and external self-play processes for accurate medical diagnostics. Self-Talk (Ulmer et al., 2024) generates role-playing dialogue data via role simulations, further processed through automated quality filtering. Sotopia-π (Wang et al., 2024c) uses GPT-4 (OpenAI, 2024a) to create varied social scenarios and objectives, further extending social task generation.

Self-Refinement employs an iterative process where models refine their outputs through self-feedback, progressively enhancing response quality. SCORE (Zhang et al., 2024f) facilitates self-correction in small models using correct solutions as feedback prompts during generation. SELF (Lu et al.) introduces a meta-skill learning framework, enabling models to improve through self-evaluation of unlabelled instructions iteratively. SELF-ALIGN (Sun et al., 2023) ensures reliable response through continuous refinement based on principles and demonstrations. ISR-LLM (Zhou et al., 2023d) employs LLMs as self-verifiers, providing feedback for iterative plan refinement.

Weak Supervision explores the potential of LLMs learning from outputs generated by weaker models. This approach addresses the growing disparity between model capabilities and human supervision limits. (Hase et al., 2024) illustrate LLMs' capacity to generalize from simple to complex tasks, validating weak supervision's feasibility. (Burns et al., 2023) focus on fine-tuning strong models using weak model outputs, introducing a performance gap recovered (PGR) metric for evaluating weak-to-strong generalization. (Bansal et al., 2024) challenge the superiority of strong but expensive models, demonstrating that weak but cheap models may offer higher computational efficiency.

Takeaways 1 *We categorize scalable SFT methods into instruction and response approaches. For instructions, context-based methods enrich instructions with external knowledge but require reliable sources, while evolution-based methods automate instruction refinement but demand careful design. For responses, sampling achieves optimal distribution but may produce redundant outputs; self-play and self-refinement iteratively boost model performance through self-feedback but are limited by the model's initial capability. Weak supervision trains stronger models using outputs from weaker ones, enabling strong generalization but with the risk of propagating errors.*

4 Scaling for Reinforcement Learning

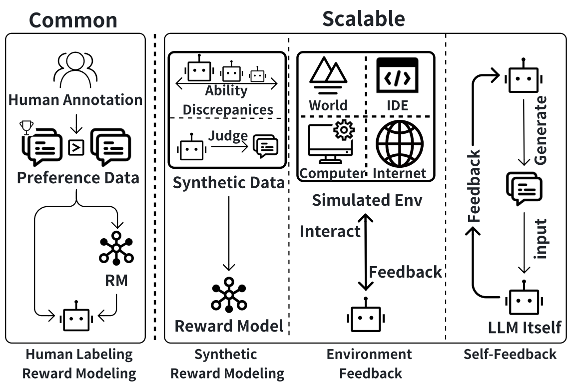

Figure 4: Classification of RL Methods.

Reinforcement Learning (RL) refers to learning from environments through interaction and rewards. Integrating RL with LLMs has become a promising area of research. Notably, InstructGPT (Ouyang et al., 2022) introduces Reinforcement Learning from Human Feedback (RLHF), enabling LLMs to understand human preferences via feedback, a technique foundational to ChatGPT (OpenAI, 2022). Several studies (Yuan et al., 2023d; Dong et al., 2023; Lee et al., 2024) aim to enhance RLHF. Scaling RL is another critical focus, as learning from environment feedback is more complex but scalable than SFT. Facilitating LLMs' training through feedback is vital for advancing scaling laws. We classify existing practices combining RL and LLMs by feedback signals (Figure 4) and analyze the scalability prospects of various RL for LLM algorithms.

4.1 The Common Methods

Reinforcement Learning within LLMs traditionally serve as supplementary alignment techniques, such as in RLHF (Ouyang et al., 2022; Li et al., 2023b; Hu et al., 2024). Consequently, considerations of scalability are often overlooked. These approaches are usually implemented as incremental post-pretraining stages, using **human labeling** or **reward modeling** to fine-tune models' outputs.

4.1.1 Human Labeling

Employing experts to provide direct alignment signals is intuitive but poses scalability challenges due to the time and cost required. Studies leverage human feedback or existing datasets for aligning LLMs, such as DPO (Rafailov et al., 2024), which optimizes reward maximization within a single policy phase, and KTO (Ethayarajh et al., 2024), which uses human-aware loss functions to eliminate the need for preference data. RRHF (Yuan et al., 2023c) aligns model responses with human preferences through a ranking-based approach, and SimPO (Meng et al., 2024) enhances this by using length-normalized rewards and target reward differences, outperforming similar methods without additional reference models.

4.1.2 Reward Modeling

The integration of model-based reward signals can significantly enhance the scalability of RLHF, yet still needs lots of human labeling. InstructGPT (Ouyang et al., 2022) integrates supervised policy training with human feedback, optimizing through Proximal Policy Optimization (PPO) (Schulman et al., 2017). ReMax (Li et al., 2024d), leveraging the REINFORCE algorithm (Williams, 1987), is more computationally efficient, and Fine-Grained Human Feedback (Wu et al., 2023b) provides detailed feedback using several reward models. Both ORM and PRM (Cobbe et al., 2021; Lightman et al., 2023) boost performance in mathematical tasks with human labeling. WARM (Ramé et al., 2024) mitigates reward hacking by fine-tuning and averaging multiple reward models. UNA (Wang et al., 2025) unifies RLHF into a supervised learning problem through a generalized implicit reward function, reducing training instability and memory requirements.

4.2 The Scalable Methods

As scaling pre-trained models hits its limits, LLM development increasingly focuses on boosting performance via Reinforcement Learning (Bai et al., 2024; Putta et al., 2024; Yuan et al., 2024). This involves automating feedback acquisition and improving alignment with human expectations. Scalable RL methods are categorized by the type of feedback signal: **synthetic reward**, **environment**, and **self**.

4.2.1 Synthetic Reward Modeling

It emphasizes scalability and leverages synthetic data and iterative processes, thus reducing reliance on manual annotations for reward modeling. RLAIF (Bai et al., 2022; Lee et al., 2024) trains preference models with AI-generated feedback from constitutions, while RLSF (Kim et al., 2023) uses quality discrepancies among LLM responses to train a reward model. Q* (Wang et al., 2024a) optimizes state prioritization using historical and future rewards, and IterAlign (Chen et al., 2024b) employs a red team approach for self-alignment and constitution discovery. Additionally, (Sun et al., 2024) implements easy-to-hard generator generalization through evaluators trained under easy task supervision.

4.2.2 Environment Feedback

It is a key reward signal for training LLMs as agents in various settings from real-world simulations (Shridhar et al., 2021) to digital environments (Zhou et al., 2024a; Rawles et al., 2024a) and rule-based systems (Côté et al., 2019). This feedback reduces the need for manually labeled data and provides consistent, reliable signals if simulations closely mimic real-world scenarios. Notable contributions include DigiRL (Bai et al., 2024), creating a parallel GUI learning environment, and ENVISIONS (Xu et al., 2024a), featuring LLM-generated trajectories interacting with simulations. Additionally, SANDBOX (Liu et al., 2023b) and RLTF (Liu et al., 2023a) explore interactive feedback and unit tests, while language-rule environments employ innovative methods like SPAG (Cheng et al., 2024b) and Prover-Verifier Games (Kirchner et al., 2024) for training models through adversarial and verification tasks.

4.2.3 Self-Feedback

It evaluates policy trajectories to enhance the generation capabilities of LLMs using the generation-discrimination gap. Techniques like RLCD (Yang et al., 2024c), Agent Q (Putta et al., 2024), and Self-Rewarding (Yuan et al., 2024) use comparative outputs, guided MCTS, and self-evaluation. The Meta-Rewarding framework (Wu et al., 2024a) has models that assess their evaluations to improve both generative and evaluative abilities. The Self-Taught Evaluator (Wang et al., 2024d) refines models by iteratively modifying input prompts, achieving superior performance on benchmarks like Reward-Bench (Lambert et al., 2024b).

Takeaways 2 *We categorize scalable RL methods based on the source of reward signals: Synthetic Reward Modeling constructs signals through rules, but it may have biases; Environment Feedback involves extensive interactions and feedback within virtual environments, but the construction cost is high, and it may not align with real-world scenarios; Self-Feedback uses self-evaluation as a signal, making it the easiest to scale, but the model's capabilities limit it.*

5 Scaling for Test-time Compute

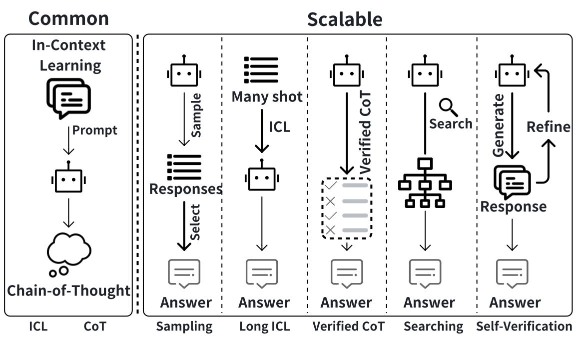

Figure 5: Classification of TTC Methods.

Test-time Compute refers to a model's inference phase, predicting outputs like the next word in a sentence. Scaling Test-time Compute, introduced by OpenAI's o1 (OpenAI, 2024b), can further enhance model performance in various scenarios. This section explores common and scalable methods for improving model performance by increasing inference computation (Figure 5).

5.1 The Common Methods

Prior research aims to enhance model performance by incorporating additional computational processes during inference. However, many of them do not consider scalability. The common methods can be divided into two categories: **In-Context Learning (ICL)** and **Chain-of-Thought (CoT)**.

5.1.1 In-Context Learning

Exemplified by GPT-3 (Brown, 2020), ICL enables LLMs to adapt to specific tasks without parameter updates by providing predefined context. Early manual prompt engineering approaches (Petroni et al., 2019; Schick and Schütze, 2020) are not scalable or automated. Recent advancements, such as automated prompt learning methods like instruction induction (Honovich et al., 2022b), APE (Zhou et al., 2022), and OPRO (Yang et al., 2024b), address some of these issues but still face challenges in performance scaling with increased input tokens during inference.

5.1.2 Chain-of-Thought

CoT (Wei et al., 2023a) enables LLMs to reason, enhances performance in logic and calculations, and offers greater flexibility and scalability. Research in CoT includes using prompting (Kojima et al., 2023; Zhou et al., 2023b; Khot et al., 2022) and training (Lewkowycz et al., 2022) to stimulate reasoning capabilities of LLMs. CoT can also integrate with tools to enhance performance in domains like mathematics and coding (Gao et al., 2023; Chen et al., 2023a) through approaches like ReAct (Yao et al., 2022b) and Inner Monologue (Huang et al., 2022b).

5.2 The Scalable Methods

In the post-GPT era, as model sizes grow, increasing parameters becomes challenging. Thus, attention shifts to scaling TTC. (Snell et al., 2024) explores inference-time computation scalability in LLMs, proposing a "compute-optimal" scaling strategy. (Li et al., 2024c) shows that TTC can significantly enhance models' expressive capabilities, demonstrating the feasibility of addressing larger-scale problems with adequate computational resources. This section categorizes various methods for augmenting computation.

5.2.1 Sampling

Sampling is a fundamental technique to boost performance by increasing test-time computation. Given the inherently stochastic nature of contemporary LLM generation strategies (Chen et al., 2021a; Holtzman et al., 2020; Zhu et al., 2023), selecting the correct answer from multiple samples for a single query enhances performance, with improvements scaling with the number of samples (Chen et al., 2021b; Wang et al., 2023a). Early approaches like Self-Consistency (Wang et al., 2023a) improve performance in mathematical domains by generating multiple reasoning paths and selecting the majority answer. Verification-guided weighted majority voting further enhances models like GPT-4 in solving mathematical problems (Zhou et al., 2023a). Additionally, (Liu et al., 2023c; Nakano et al., 2022; Cobbe et al., 2021) employs reward models to select human-aligned responses and solve mathematical problems.

5.2.2 Verified Chain-of-Thought

Verifying reasoning paths is critical as the correctness of each inferential step significantly impacts LLM performance (Weng et al., 2023). Various methods propose enhancing the reliability and scalability of CoT reasoning. PRM (Lightman et al., 2023) verify each reasoning step, outperforming majority voting. SelfCheck (Miao et al., 2023) allows for step verification to identify and regenerate erroneous steps, improving long-chain reliability. DiVeRSe (Li et al., 2023a) uses diverse prompts and weighted voting to enhance performance. LogiCoT (Zhao et al., 2024b) integrates symbolic logic to verify and adjust reasoning processes, mitigating hallucinations due to logical errors.

5.2.3 Searching

Searching utilizes different reasoning structures to explore in space. Tree-of-Thought (ToT) (Yao et al., 2023) self-evaluates and backtracks multiple reasoning paths, enhancing problem-solving. Diverse search algorithms (Golovneva et al., 2023; Zhang et al., 2023b, 2024e; Sel et al., 2024) like Pathfinder (Golovneva et al., 2023), Cumulative Reasoning (Zhang et al., 2023b), Graph of Thoughts (Besta et al., 2024), and Diagram of Thought (Zhang et al., 2024e) optimize tree searches, iterative methods, graph models, and directed acyclic graphs, respectively. Algorithm of Thoughts (Sel et al., 2024) further optimizes ToT algorithms for computational efficiency, and MCTS explores boosting test-time performance (Liu et al., 2024a; Feng et al., 2024; Tian et al., 2024; Zhang et al., 2024b).

5.2.4 Long In-Context Learning

With the ongoing research in automated ICL, performance improvements become unstable as the number of examples increases, thereby underutilizing the potential of multi-instance contextual learning (Lu et al., 2022). Consequently, Many-Shot ICL (Agarwal et al., 2024) devises an automated example construction technique, extending examples to up to 2048 shots, leading to noticeable advancements across various domains. Another approach (Hsieh et al., 2023) combines a greedy algorithm with beam search to optimize the method for long prompts, thereby enhancing the scaling performance of long in-context learning.

5.2.5 Self-Verification

Enabling LLMs to self-verify at test time improves performance by bridging the gap between generation and discrimination (Ng and Jordan, 2001; Zheng et al., 2023). Self-Refine (Press et al., 2022) uses iterative self-feedback to enhance outputs, while Reflexion (Shinn et al., 2024) mitigates hallucinations and inefficiencies through environmental feedback. (Li et al., 2024a) propose a self-checking paradigm for comparing candidate answers, and Self-Verification (Weng et al., 2023) provides an explainable validation score via reverse verification. Self-Contrast (Zhang et al., 2024d) generates and compares different solutions to avoid biases and inconsistencies during re-evaluation.

Takeaways 3 *We categorize scalable TTC methods as follows: Sampling is the simplest scaling method but may produce redundant similar output. Verified CoT aims to address the scalability issue limited by error propagation in CoT. Searching aims to explore optimal paths in the reasoning space, utilizing various reasoning structures. Long ICL enhances model output through many unsupervised examples. Self-verification improves performance through self-correction. These meth-*

ods exchange model performance with inference computation.

6 Potential Applications

As LLMs continue improving, research has shifted from chat-based applications to productivity-based ones. Post-training efforts now focus on enhancing these productivity applications. This section discusses these emerging, promising applications.

6.1 Mathematics

LLMs have long struggled with Mathematics, but recent advancements are changing this landscape. Efforts to scale training data include using extensive datasets from the web (Lewkowycz et al., 2022; Taylor et al., 2022; Yue et al., 2024; Toshniwal et al., 2024a). Some works focus on generating CoT training data (Zelikman et al., 2022; Wang et al., 2024b; Toshniwal et al., 2024b; Qiao et al., 2024) and using RMs to select high-quality data (Luo et al., 2023; Xu et al., 2024b; Yang et al., 2024a; Cobbe et al., 2021). Other works enhance mathematical reasoning using self-generated data. For instance, LMSI (Huang et al., 2022a) uses high-confidence responses, while RFT (Yuan et al., 2023b) applies correct answers for filtering. RL also strengthens LLMs in math. PRM (Lightman et al., 2023) introduces step supervision, and DeepSeek-Math improves PPO efficiency (Shao et al., 2024). Math-Shepherd (Wang et al., 2024b) implements step-PPO via PRM, whereas Step-DPO (Lai et al., 2024b) focuses on individual reasoning steps for DPO. Lastly, OmegaPRM (Luo et al., 2024b) uses a divide-and-conquer style MCTS to gather high-quality supervision data.

6.2 Code Generation

Despite the advancements in LLMs' code generation abilities, they still struggle with complex engineering problems (Chen et al., 2021c; Li et al., 2022; Fried et al., 2022; Lai et al., 2023; Nijkamp et al., 2022). Scalable post-training techniques are needed, with a key aspect being feedback from the runtime environment for optimization. One approach involves using external tools like code interpreters, as seen in LDB (Zhong et al., 2024) and SelfEvolve (Jiang et al., 2023), which help LLMs execute code and obtain error feedback. SelfDebugging (Chen et al., 2023b) focuses on optimizing code through feedback from execution outcomes. For repository-level code, RepoCoder (Zhang et al., 2023a) uses a retrieval framework to extract valuable information from repositories during generation. ARKS (Su et al., 2024) creates a "knowledge soup" to improve code generation iteratively. RLEF (Gehring et al., 2024) proposes an RL approach to enhance LLMs in code synthesis by utilizing execution feedback to improve code iteratively.

6.3 Agent Execution

The growing capabilities of LLMs extend beyond text generation and program synthesis to include device control in simulated environments like web and android platforms (Bai et al., 2024; Lai et al., 2024a; Zhou et al., 2023c; Zhang et al., 2024c; Xu et al., 2024c; Rawles et al., 2024b; Deng et al., 2024; Yao et al., 2022a). Research involves searching and RL strategies, which generate trajectories through online interaction and are evaluated by reward models (Pan et al., 2024; Qi et al., 2024; Liu et al., 2024b). DigiRL (Bai et al., 2024) employs online learning and AWR for policy updates, excelling on the AITW benchmark (Rawles et al., 2024b). Archer (Zhou et al., 2024b) uses hierarchical RL for multi-round decision-making, while AgentQ (Putta et al., 2024) leverages MCTS and DPO for searching and policy updates (Rafailov et al., 2024).

7 Conclusion

The landscape of LLM training is evolving as traditional pre-training scaling shows limitations. This survey examines the new paradigm of post-training scaling, specifically SFT, RLxF, and TTC. By comparing these methods and highlighting their potential to overcome computational and data limitations, we provide insights for future research, setting the stage for integrating them into more efficient and sustainable advancements in LLMs.

Acknowledgement

This work has been supported by the NSFC (624B2084), NSFC for Distinguished Young Scholar (62425601) and New Cornerstone Science Foundation through the XPLORER PRIZE.

Limitations

This section discusses several challenges and future directions of the post-training scaling in LLMs.

Theoretical Foundation. Existing methods for scaling post-training rely heavily on empirical experience and need a theoretical foundation. Some research efforts focus on theoretical analysis. (Li et al., 2024c) provided theoretical evidence for TTC by demonstrating how it enhances the sequential computation capability of Transformer architectures, addressing their inherent limitations in handling serial reasoning tasks. (Snell et al., 2024) examined the theoretical support of self-improvement mechanisms, explaining how the revision-and-refine approach modifies implicit input distributions to enhance model reasoning. (Ngo et al., 2024) developed a theoretical framework for analyzing alignment problems, clarifying how RLHF mitigates distribution shifts to improve performance. Additionally, (Dai et al., 2023) provided theoretical insights into instruction tuning from a gradient descent perspective, explaining how it implicitly optimizes models to acquire in-context learning abilities.

Synthetic Data. To support the scaling of post-training, constructing high-quality synthetic data is essential. Although synthetic data have shown considerable success in advancing model abilities, several studies have highlighted the dark side of synthetic data, including model collapse (Shumailov et al., 2024). Furthermore, ensuring data diversity presents a significant challenge. Without human intervention, augmented data often do not exceed the distribution of the initial seed set. Thus, synthesizing high-quality data while mitigating potential risks remains an open problem.

Continual Learning. The ideal approach for post-training involves continuously collecting data from dynamic environments, such as the real world, to incrementally enhance LLM's performance. The primary objective is to enable the model to acquire new skills and knowledge over time while preserving its existing capabilities. In the context of scaling post-training for LLMs, addressing the alignment tax (Ouyang et al., 2022) and mitigating the forgetting of acquired knowledge is crucial.

Active Exploration. Current post-training methods rely on manually curated or model-augmented data derived from a human-collected seed set. This can introduce biases due to intrinsic human knowledge limitations, and it remains unclear whether these human-inspected datasets are practical for scaling. Another approach involves empowering the model to identify underperforming areas and actively generate targeted data for enhancement. Recent studies have explored the utilization of advanced LLMs as teachers, dynamically assessing the performance of a student model and generating specific training samples accordingly (Cheng et al., 2024a; Lu et al., 2024a). However, challenges arise in self-evolution scenarios. For example, self-evaluation bias can present significant difficulties, as an accurate assessment of the model's performance is crucial before generating supplementary training data. Future research should focus on enabling models to actively self-discover effective training samples, thereby facilitating their autonomous learning processes.

Superalignment. The base model could become incredibly powerful as we scale the model size and data. However, the methodologies to effectively conduct post-training to fully harness these capabilities still need to be explored, also known as the weak-to-strong generalization problem (Burns et al., 2023). Moreover, significant efforts must be dedicated to safety post-training to ensure these models do not pose any risks. It is crucial to discern whether a model is genuinely safe or merely simulating safe behavior (Wang et al., 2023b).

Evaluation Metrics and Benchmarks. The evaluation of post-training effectiveness has traditionally depended on static benchmarks. However, with the increasing capabilities of LLMs, it is imperative to design more challenging and comprehensive benchmarks. Additionally, issues such as data leakage (Yang et al., 2023; Wei et al., 2023b) and leaderboard saturation (Guo et al., 2023) have become prevalent. To address these challenges, it is essential to innovate new evaluation methods, such as dynamic and automated leaderboards, and to develop novel metrics that effectively assess the impact of scaling.

References

Rishabh Agarwal, Avi Singh, Lei M. Zhang, Bernd Bohnet, Luis Rosias, Stephanie Chan, Biao Zhang, Ankesh Anand, Zaheer Abbas, Azade Nova, John D. Co-Reyes, Eric Chu, Feryal Behbahani, Aleksandra Faust, and Hugo Larochelle. 2024. Many-shot in-context learning.

Arian Askari, Roxana Petcu, Chuan Meng, Mohammad Aliannejadi, Amin Abolghasemi, Evangelos Kanoulas, and Suzan Verberne. 2024. Self-seeding and multi-intent self-instructing llms for generating intent-aware information-seeking dialogs.

Stephen Bach, Victor Sanh, Zheng Xin Yong, Albert Webson, Colin Raffel, Nihal V. Nayak, Abheesht Sharma, Taewoon Kim, M Saiful Bari, Thibault Fevry, Zaid Alyafeai, Manan Dey, Andrea Santilli, Zhiqing Sun, Srulik Ben-david, Canwen Xu, Gunjan Chhablani, Han Wang, Jason Fries, Maged Alshaibani, Shanya Sharma, Urmish Thakker, Khalid Almubarak, Xiangru Tang, Dragomir Radev, Mike Tian-jian Jiang, and Alexander Rush. 2022. PromptSource: An integrated development environment and repository for natural language prompts. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 93–104, Dublin, Ireland. Association for Computational Linguistics.

Hao Bai, Yifei Zhou, Mert Cemri, Jiayi Pan, Alane Suhr, Sergey Levine, and Aviral Kumar. 2024. Digirl: Training in-the-wild device-control agents with autonomous reinforcement learning.

Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, Carol Chen, Catherine Olsson, Christopher Olah, Danny Hernandez, Dawn Drain, Deep Ganguli, Dustin Li, Eli Tran-Johnson, Ethan Perez, Jamie Kerr, Jared Mueller, Jeffrey Ladish, Joshua Landau, Kamal Ndousse, Kamile Lukosuite, Liane Lovitt, Michael Sellitto, Nelson Elhage, Nicholas Schiefer, Noemi Mercado, Nova DasSarma, Robert Lasenby, Robin Larson, Sam Ringer, Scott Johnston, Shauna Kravec, Sheer El Showk, Stanislav Fort, Tamera Lanham, Timothy Telleen-Lawton, Tom Conerly, Tom Henighan, Tristan Hume, Samuel R. Bowman, Zac Hatfield-Dodds, Ben Mann, Dario Amodei, Nicholas Joseph, Sam McCandlish, Tom Brown, and Jared Kaplan. 2022. Constitutional ai: Harmlessness from ai feedback.

Hritik Bansal, Arian Hosseini, Rishabh Agarwal, Vinh Q Tran, and Mehran Kazemi. 2024. Smaller, weaker, yet better: Training llm reasoners via compute-optimal sampling. *arXiv preprint arXiv:2408.16737*.

Emily M Bender, Timnit Gebru, Angelina McMillan-Major, and Shmargaret Shmitchell. 2021. On the dangers of stochastic parrots: Can language models be too big? In *Proceedings of the 2021 ACM conference on fairness, accountability, and transparency*, pages 610–623.

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and Torsten Hoefler. 2024. Graph of thoughts: Solving elaborate problems with large language models. *Proceedings of the AAAI Conference on Artificial Intelligence*, 38(16):17682–17690.

Tom B Brown. 2020. Language models are few-shot learners. *arXiv preprint arXiv:2005.14165*.

Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. 2020. Language models are few-shot learners.

Alexander Bukharin, Yixiao Li, Pengcheng He, and Tuo Zhao. 2024. Deep reinforcement learning from hierarchical preference design.

Collin Burns, Pavel Izmailov, Jan Hendrik Kirchner, Bowen Baker, Leo Gao, Leopold Aschenbrenner, Yining Chen, Adrien Ecoffet, Manas Joglekar, Jan Leike, Ilya Sutskever, and Jeff Wu. 2023. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021a. Evaluating large language models trained on code.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan

Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021b. Evaluating large language models trained on code.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. 2021c. Evaluating large language models trained on code.

Pinzhen Chen, Zhicheng Guo, Barry Haddow, and Kenneth Heafield. 2024a. Iterative translation refinement with large language models.

Wenhu Chen, Xueguang Ma, Xinyi Wang, and William W. Cohen. 2023a. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks.

Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023b. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128*.

Xiusi Chen, Hongzhi Wen, Sreyashi Nag, Chen Luo, Qingyu Yin, Ruirui Li, Zheng Li, and Wei Wang. 2024b. Iteralign: Iterative constitutional alignment of large language models.

Zixiang Chen, Yihe Deng, Huizhuo Yuan, Kaixuan Ji, and Quanquan Gu. 2024c. Self-play fine-tuning converts weak language models to strong language models.

Jiale Cheng, Yida Lu, Xiaotao Gu, Pei Ke, Xiao Liu, Yuxiao Dong, Hongning Wang, Jie Tang, and Minlie Huang. 2024a. Autodetect: Towards a unified framework for automated weakness detection in large language models. *arXiv preprint arXiv:2406.16714*.

Pengyu Cheng, Tianhao Hu, Han Xu, Zhisong Zhang, Yong Dai, Lei Han, and Nan Du. 2024b. Self-playing adversarial language game enhances llm reasoning.

Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. 2023. Palm: Scaling language modeling with pathways. *Journal of Machine Learning Research*, 24(240):1–113.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems.

Ganqu Cui, Lifan Yuan, Ning Ding, Guanming Yao, Wei Zhu, Yuan Ni, Guotong Xie, Zhiyuan Liu, and Maosong Sun. 2023. Ultrafeedback: Boosting language models with high-quality feedback. *arXiv preprint arXiv:2310.01377*.

Wanyun Cui and Qianle Wang. 2023. Ada-instruct: Adapting instruction generators for complex reasoning.

Marc-Alexandre Côté, Ákos Kádár, Xingdi Yuan, Ben Kybartas, Tavian Barnes, Emery Fine, James Moore, Ruo Yu Tao, Matthew Hausknecht, Layla El Asri, Mahmoud Adada, Wendy Tay, and Adam Trischler. 2019. Textworld: A learning environment for text-based games.

Damai Dai, Yutao Sun, Li Dong, Yaru Hao, Shuming Ma, Zhifang Sui, and Furu Wei. 2023. Why can gpt learn in-context? language models implicitly perform gradient descent as meta-optimizers.

Xiang Deng, Yu Gu, Boyuan Zheng, Shijie Chen, Sam Stevens, Boshi Wang, Huan Sun, and Yu Su. 2024. Mind2web: Towards a generalist agent for the web. *Advances in Neural Information Processing Systems*, 36.

Ruomeng Ding, Chaoyun Zhang, Lu Wang, Yong Xu, Minghua Ma, Wei Zhang, Si Qin, Saravan Rajmohan, Qingwei Lin, and Dongmei Zhang. 2024. Everything of thoughts: Defying the law of penrose triangle for thought generation.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Yihan Zhang, Winnie Chow, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment.

Dheeru Dua, Yizhong Wang, Pradeep Dasigi, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Drop: A reading comprehension benchmark requiring discrete reasoning over paragraphs.

Kawin Ethayarajh, Winnie Xu, Niklas Muennighoff, Dan Jurafsky, and Douwe Kiela. 2024. Kto: Model alignment as prospect theoretic optimization.

Xidong Feng, Ziyu Wan, Muning Wen, Stephen Marcus McAleer, Ying Wen, Weinan Zhang, and Jun Wang. 2024. Alphazero-like tree-search can guide large language model decoding and training.

Chrisantha Fernando, Dylan Banarse, Henryk Michalewski, Simon Osindero, and Tim Rocktäschel. 2023. Promptbreeder: Self-referential self-improvement via prompt evolution.

Daniel Fried, Armen Aghajanyan, Jessy Lin, Sida Wang, Eric Wallace, Freda Shi, Ruiqi Zhong, Wen-tau Yih, Luke Zettlemoyer, and Mike Lewis. 2022. Incoder: A generative model for code infilling and synthesis. *arXiv preprint arXiv:2204.05999*.

Leo Gao, Stella Biderman, Sid Black, Laurence Golding, Travis Hoppe, Charles Foster, Jason Phang, Horace He, Anish Thite, Noa Nabeshima, et al. 2020. The pile: An 800gb dataset of diverse text for language modeling. *arXiv preprint arXiv:2101.00027*.

Luyu Gao, Aman Madaan, Shuyan Zhou, Uri Alon, Pengfei Liu, Yiming Yang, Jamie Callan, and Graham Neubig. 2023. Pal: Program-aided language models.

Jonas Gehring, Kunhao Zheng, Jade Copet, Vegard Mella, Taco Cohen, and Gabriel Synnaeve. 2024. Rlef: Grounding code llms in execution feedback with reinforcement learning.

Olga Golovneva, Sean O'Brien, Ramakanth Pasunuru, Tianlu Wang, Luke Zettlemoyer, Maryam Fazel-Zarandi, and Asli Celikyilmaz. 2023. Pathfinder: Guided search over multi-step reasoning paths.

Xinyan Guan, Yanjiang Liu, Xinyu Lu, Boxi Cao, Ben He, Xianpei Han, Le Sun, Jie Lou, Bowen Yu, Yaojie Lu, and Hongyu Lin. 2024. Search, verify and feedback: Towards next generation post-training paradigm of foundation models via verifier engineering.

Zishan Guo, Renren Jin, Chuang Liu, Yufei Huang, Dan Shi, Linhao Yu, Yan Liu, Jiaxuan Li, Bojian Xiong, Deyi Xiong, et al. 2023. Evaluating large language models: A comprehensive survey. *arXiv preprint arXiv:2310.19736*.

Peter Hase, Mohit Bansal, Peter Clark, and Sarah Wiegreffe. 2024. The unreasonable effectiveness of easy training data for hard tasks.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, et al. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2020. The curious case of neural text degeneration.

Or Honovich, Thomas Scialom, Omer Levy, and Timo Schick. 2022a. Unnatural instructions: Tuning language models with (almost) no human labor.

Or Honovich, Uri Shaham, Samuel R Bowman, and Omer Levy. 2022b. Instruction induction: From few examples to natural language task descriptions. *arXiv preprint arXiv:2205.10782*.

Cho-Jui Hsieh, Si Si, Felix X. Yu, and Inderjit S. Dhillon. 2023. Automatic engineering of long prompts.

Jian Hu, Xibin Wu, Weixun Wang, Xianyu, Dehao Zhang, and Yu Cao. 2024. Openrlhf: An easy-to-use, scalable and high-performance rlhf framework.

Jiaxin Huang, Shixiang Shane Gu, Le Hou, Yuexin Wu, Xuezhi Wang, Hongkun Yu, and Jiawei Han. 2022a. Large language models can self-improve.

Wenlong Huang, Fei Xia, Ted Xiao, Harris Chan, Jacky Liang, Pete Florence, Andy Zeng, Jonathan Tompson, Igor Mordatch, Yevgen Chebotar, et al. 2022b. Inner monologue: Embodied reasoning through planning with language models. *arXiv preprint arXiv:2207.05608*.

Shuyang Jiang, Yuhao Wang, and Yu Wang. 2023. Self-evolve: A code evolution framework via large language models.

Tushar Khot, Harsh Trivedi, Matthew Finlayson, Yao Fu, Kyle Richardson, Peter Clark, and Ashish Sabharwal. 2022. Decomposed prompting: A modular approach for solving complex tasks. *arXiv preprint arXiv:2210.02406*.

Sungdong Kim, Sanghwan Bae, Jamin Shin, Soyoung Kang, Donghyun Kwak, Kang Min Yoo, and Minjoon Seo. 2023. Aligning large language models through synthetic feedback.

Jan Hendrik Kirchner, Yining Chen, Harri Edwards, Jan Leike, Nat McAleese, and Yuri Burda. 2024. Prover-verifier games improve legibility of llm outputs.

Denis Kocetkov, Raymond Li, Loubna Ben Allal, Jia Li, Chenghao Mou, Carlos Muñoz Ferrandis, Yacine Jernite, Margaret Mitchell, Sean Hughes, Thomas Wolf, Dzmitry Bahdanau, Leandro von Werra, and Harm de Vries. 2022. The stack: 3 tb of permissively licensed source code. *Preprint*.

Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. 2023. Large language models are zero-shot reasoners.

Aviral Kumar, Vincent Zhuang, Rishabh Agarwal, Yi Su, John D Co-Reyes, Avi Singh, Kate Baumli, Shariq Iqbal, Colton Bishop, Rebecca Roelofs, Lei M Zhang, Kay McKinney, Disha Shrivastava, Cosmin Paduraru, George Tucker, Doina Precup, Feryal Behbahani, and Aleksandra Faust. 2024. Training language models to self-correct via reinforcement learning.

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

Hanyu Lai, Xiao Liu, Iat Long Iong, Shuntian Yao, Yuxuan Chen, Pengbo Shen, Hao Yu, Hanchen Zhang, Xiaohan Zhang, Yuxiao Dong, et al. 2024a. Autowebglm: Bootstrap and reinforce a large language model-based web navigating agent. *arXiv preprint arXiv:2404.03648*.

Xin Lai, Zhuotao Tian, Yukang Chen, Senqiao Yang, Xiangru Peng, and Jiaya Jia. 2024b. Step-dpo: Step-wise preference optimization for long-chain reasoning of llms.

Yuhang Lai, Chengxi Li, Yiming Wang, Tianyi Zhang, Ruiqi Zhong, Luke Zettlemoyer, Wen-tau Yih, Daniel Fried, Sida Wang, and Tao Yu. 2023. Ds-1000: A natural and reliable benchmark for data science code generation. In *International Conference on Machine Learning*, pages 18319–18345. PMLR.

Nathan Lambert, Jacob Morrison, Valentina Pyatkin, Shengyi Huang, Hamish Ivison, Faeze Brahman, Lester James V. Miranda, Alisa Liu, Nouha Dziri, Shane Lyu, Yuling Gu, Saumya Malik, Victoria Graf, Jena D. Hwang, Jiangjiang Yang, Ronan Le Bras, Oyvind Tafjord, Chris Wilhelm, Luca Soldaini, Noah A. Smith, Yizhong Wang, Pradeep Dasigi, and Hannaneh Hajishirzi. 2024a. Tulu 3: Pushing frontiers in open language model post-training.

Nathan Lambert, Valentina Pyatkin, Jacob Morrison, LJ Miranda, Bill Yuchen Lin, Khyathi Chandu, Nouha Dziri, Sachin Kumar, Tom Zick, Yejin Choi, Noah A. Smith, and Hannaneh Hajishirzi. 2024b. Rewardbench: Evaluating reward models for language modeling.

Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. 2023. Bloom: A 176b-parameter open-access multilingual language model.

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu, Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, and Sushant Prakash. 2024. Rlaif vs. rlhf: Scaling reinforcement learning from human feedback with ai feedback.

Aitor Lewkowycz, Anders Andreassen, David Dohan, Ethan Dyer, Henryk Michalewski, Vinay Ramasesh, Ambrose Slone, Cem Anil, Imanol Schlag, Theo Gutman-Solo, Yuhuai Wu, Behnam Neyshabur, Guy Gur-Ari, and Vedant Misra. 2022. Solving quantitative reasoning problems with language models.

Moxin Li, Wenjie Wang, Fuli Feng, Fengbin Zhu, Qifan Wang, and Tat-Seng Chua. 2024a. Think twice before trusting: Self-detection for large language models through comprehensive answer reflection.

Xian Li, Ping Yu, Chunting Zhou, Timo Schick, Omer Levy, Luke Zettlemoyer, Jason Weston, and Mike Lewis. 2024b. Self-alignment with instruction back-translation.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023a. Making large language models better reasoners with step-aware verifier.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, et al. 2022. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097.

Zhiyuan Li, Hong Liu, Denny Zhou, and Tengyu Ma. 2024c. Chain of thought empowers transformers to solve inherently serial problems. *arXiv preprint arXiv:2402.12875*.

Zihao Li, Zhuoran Yang, and Mengdi Wang. 2023b. Reinforcement learning with human feedback: Learning dynamic choices via pessimism.

Ziniu Li, Tian Xu, Yushun Zhang, Zhihang Lin, Yang Yu, Ruoyu Sun, and Zhi-Quan Luo. 2024d. Remax: A simple, effective, and efficient reinforcement learning method for aligning large language models.

Zhenwen Liang, Ye Liu, Tong Niu, Xiangliang Zhang, Yingbo Zhou, and Semih Yavuz. 2024. Improving llm reasoning through scaling inference computation with collaborative verification.

Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. 2023. Let's verify step by step.

Jiacheng Liu, Andrew Cohen, Ramakanth Pasunuru, Yejin Choi, Hannaneh Hajishirzi, and Asli Celikyilmaz. 2024a. Don't throw away your value model! generating more preferable text with value-guided monte-carlo tree search decoding.

Jiate Liu, Yiqin Zhu, Kaiwen Xiao, Qiang Fu, Xiao Han, Wei Yang, and Deheng Ye. 2023a. Rltf: Reinforcement learning from unit test feedback.

Ruibo Liu, Ruixin Yang, Chenyan Jia, Ge Zhang, Denny Zhou, Andrew M. Dai, Diyi Yang, and Soroush Vosoughi. 2023b. Training socially aligned language models on simulated social interactions.

Xiao Liu, Hanyu Lai, Hao Yu, Yifan Xu, Aohan Zeng, Zhengxiao Du, Peng Zhang, Yuxiao Dong, and Jie Tang. 2023c. Webglm: Towards an efficient web-enhanced question answering system with human preferences.

Xiao Liu, Bo Qin, Dongzhu Liang, Guang Dong, Hanyu Lai, Hanchen Zhang, Hanlin Zhao, Iat Long Iong, Jiadai Sun, Jiaqi Wang, Junjie Gao, Junjun Shan, Kangning Liu, Shudan Zhang, Shuntian Yao, Siyi Cheng, Wentao Yao, Wenyi Zhao, Xinghan Liu, Xinyi Liu, Xinying Chen, Xinyue Yang, Yang Yang, Yifan Xu, Yu Yang, Yujia Wang, Yulin Xu, Zehan Qi, Yuxiao Dong, and Jie Tang. 2024b. Autoglm: Autonomous foundation agents for guis.

Xiao Liu, Fanjin Zhang, Zhenyu Hou, Li Mian, Zhaoyu Wang, Jing Zhang, and Jie Tang. 2021. Self-supervised learning: Generative or contrastive. *IEEE transactions on knowledge and data engineering*, 35(1):857–876.

Kyle Lo, Lucy Lu Wang, Mark Neumann, Rodney Kinney, and Dan S Weld. 2019. S2orc: The semantic scholar open research corpus. *arXiv preprint arXiv:1911.02782*.

Shayne Longpre, Le Hou, Tu Vu, Albert Webson, Hyung Won Chung, Yi Tay, Denny Zhou, Quoc V. Le, Barret Zoph, Jason Wei, and Adam Roberts. 2023. The flan collection: Designing data and methods for effective instruction tuning.

Jianqiao Lu, Wanjun Zhong, Wenyong Huang, Yufei Wang, Qi Zhu, Fei Mi, Baojun Wang, Weichao Wang, Xingshan Zeng, Lifeng Shang, et al. Self: Self-evolution with language feed.

Jianqiao Lu, Wanjun Zhong, Yufei Wang, Zhijiang Guo, Qi Zhu, Wenyong Huang, Yanlin Wang, Fei Mi, Baojun Wang, Yasheng Wang, et al. 2024a. Yoda: Teacher-student progressive learning for language models. *arXiv preprint arXiv:2401.15670*.

Keming Lu, Bowen Yu, Chang Zhou, and Jingren Zhou. 2024b. Large language models are superpositions of all characters: Attaining arbitrary role-play via self-alignment.

Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. 2022. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Qingwei Lin, Jianguang Lou, Shifeng Chen, Yansong Tang, and Weizhu Chen. 2024a. Arena learning: Build data flywheel for llms post-training via simulated chatbot arena.

Haipeng Luo, Qingfeng Sun, Can Xu, Pu Zhao, Jianguang Lou, Chongyang Tao, Xiubo Geng, Qingwei Lin, Shifeng Chen, and Dongmei Zhang. 2023. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct. *arXiv preprint arXiv:2308.09583*.

Liangchen Luo, Yinxiao Liu, Rosanne Liu, Samrat Phatale, Meiqi Guo, Harsh Lara, Yunxuan Li, Lei Shu, Yun Zhu, Lei Meng, Jiao Sun, and Abhinav Rastogi. 2024b. Improve mathematical reasoning in language models by automated process supervision.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward.

Ning Miao, Yee Whye Teh, and Tom Rainforth. 2023. Selfcheck: Using llms to zero-shot check their own step-by-step reasoning.

Thomas Moreau and Julien Audiffren. 2017. Post training in deep learning with last kernel.

Reiichiro Nakano, Jacob Hilton, Suchir Balaji, Jeff Wu, Long Ouyang, Christina Kim, Christopher Hesse, Shantanu Jain, Vineet Kosaraju, William Saunders, Xu Jiang, Karl Cobbe, Tyna Eloundou, Gretchen Krueger, Kevin Button, Matthew Knight, Benjamin Chess, and John Schulman. 2022. Webgpt: Browser-assisted question-answering with human feedback.

Andrew Ng and Michael Jordan. 2001. On discriminative vs. generative classifiers: A comparison of logistic regression and naive bayes. *Advances in neural information processing systems*, 14.

Richard Ngo, Lawrence Chan, and Sören Mindermann. 2024. The alignment problem from a deep learning perspective.

Erik Nijkamp, Bo Pang, Hiroaki Hayashi, Lifu Tu, Huan Wang, Yingbo Zhou, Silvio Savarese, and Caiming Xiong. 2022. Codegen: An open large language model for code with multi-turn program synthesis. *arXiv preprint arXiv:2203.13474*.

OpenAI. 2022. Introducing chatgpt.

OpenAI. 2024a. Gpt-4 technical report.

OpenAI. 2024b. Introducing openai o1.

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. 2022. Training language models to follow instructions with human feedback.

Jiayi Pan, Yichi Zhang, Nicholas Tomlin, Yifei Zhou, Sergey Levine, and Alane Suhr. 2024. Autonomous evaluation and refinement of digital agents. In *First Conference on Language Modeling*.

Guilherme Penedo, Hynek Kydlíček, Anton Lozhkov, Margaret Mitchell, Colin Raffel, Leandro Von Werra, Thomas Wolf, et al. 2024. The fineweb datasets: Decanting the web for the finest text data at scale. *arXiv preprint arXiv:2406.17557*.

Fabio Petroni, Tim Rocktäschel, Sebastian Riedel, Patrick Lewis, Anton Bakhtin, Yuxiang Wu, and Alexander Miller. 2019. Language models as knowledge bases? In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2463–2473, Hong Kong, China. Association for Computational Linguistics.

Ofir Press, Muru Zhang, Sewon Min, Ludwig Schmidt, Noah A Smith, and Mike Lewis. 2022. Measuring and narrowing the compositionality gap in language models. *arXiv preprint arXiv:2210.03350*.

Pranav Putta, Edmund Mills, Naman Garg, Sumeet Motwani, Chelsea Finn, Divyansh Garg, and Rafael Rafailov. 2024. Agent q: Advanced reasoning and learning for autonomous ai agents.

Zehan Qi, Xiao Liu, Iat Long Iong, Hanyu Lai, Xueqiao Sun, Wenyi Zhao, Yu Yang, Xinyue Yang, Jiadai Sun, Shuntian Yao, Tianjie Zhang, Wei Xu, Jie Tang, and Yuxiao Dong. 2024. Webrl: Training llm web agents via self-evolving online curriculum reinforcement learning.

Shuofei Qiao, Ningyu Zhang, Runnan Fang, Yujie Luo, Wangchunshu Zhou, Yuchen Eleanor Jiang, Chengfei Lv, and Huajun Chen. 2024. Autoact: Automatic agent learning from scratch via self-planning. *arXiv preprint arXiv:2401.05268*.

Yuxiao Qu, Tianjun Zhang, Naman Garg, and Aviral Kumar. 2024. Recursive introspection: Teaching language model agents how to self-improve.

Rafael Rafailov, Archit Sharma, Eric Mitchell, Christopher D Manning, Stefano Ermon, and Chelsea Finn. 2024. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36.

Pranav Rajpurkar, Robin Jia, and Percy Liang. 2018. Know what you don't know: Unanswerable questions for squad.

Alexandre Ramé, Nino Vieillard, Léonard Hussenot, Robert Dadashi, Geoffrey Cideron, Olivier Bachem, and Johan Ferret. 2024. Warm: On the benefits of weight averaged reward models.

Christopher Rawles, Sarah Clinckemaillie, Yifan Chang, Jonathan Waltz, Gabrielle Lau, Marybeth Fair, Alice Li, William Bishop, Wei Li, Folawiyo Campbell-Ajala, Daniel Toyama, Robert Berry, Divya Tyamagundlu, Timothy Lillicrap, and Oriana Riva. 2024a. Androidworld: A dynamic benchmarking environment for autonomous agents.

Christopher Rawles, Alice Li, Daniel Rodriguez, Oriana Riva, and Timothy Lillicrap. 2024b. Androidinthewild: A large-scale dataset for android device control. *Advances in Neural Information Processing Systems*, 36.

Baptiste Rozière, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, Jérémy Rapin, Artyom Kozhevnikov, Ivan Evtimov, Joanna Bitton, Manish Bhatt, Cristian Canton Ferrer, Aaron Grattafiori, Wenhan Xiong, Alexandre Défossez, Jade Copet, Faisal Azhar, Hugo Touvron, Louis Martin, Nicolas Usunier, Thomas Scialom, and Gabriel Synnaeve. 2024. Code llama: Open foundation models for code.

Arthur L Samuel. 1959. Some studies in machine learning using the game of checkers. *IBM Journal of research and development*, 3(3):210–229.

Victor Sanh, Albert Webson, Colin Raffel, Stephen H. Bach, Lintang Sutawika, Zaid Alyafeai, Antoine Chaffin, Arnaud Stiegler, Teven Le Scao, Arun Raja, Manan Dey, M Saiful Bari, Canwen Xu, Urmish Thakker, Shanya Sharma Sharma, Eliza Szczechla, Taewoon Kim, Gunjan Chhablani, Nihal Nayak, Debajyoti Datta, Jonathan Chang, Mike Tian-Jian Jiang, Han Wang, Matteo Manica, Sheng Shen, Zheng Xin Yong, Harshit Pandey, Rachel Bawden, Thomas Wang, Trishala Neeraj, Jos Rozen, Abheesht Sharma, Andrea Santilli, Thibault Fevry, Jason Alan Fries, Ryan Teehan, Tali Bers, Stella Biderman, Leo Gao, Thomas Wolf, and Alexander M. Rush. 2022. Multitask prompted training enables zero-shot task generalization.

Timo Schick and Hinrich Schütze. 2020. Few-shot text generation with pattern-exploiting training. *arXiv preprint arXiv:2012.11926*.

John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. 2017. Proximal policy optimization algorithms.

Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. 2024. Algorithm of thoughts: Enhancing exploration of ideas in large language models.

Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Mingchuan Zhang, YK Li, Yu Wu, and Daya Guo. 2024. Deepseekmath: Pushing the limits of mathematical reasoning in open language models. *arXiv preprint arXiv:2402.03300*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. 2024. Reflexion: Language agents with verbal reinforcement learning. *Advances in Neural Information Processing Systems*, 36.

Mohit Shridhar, Xingdi Yuan, Marc-Alexandre Côté, Yonatan Bisk, Adam Trischler, and Matthew Hausknecht. 2021. Alfworld: Aligning text and embodied environments for interactive learning.

Ilia Shumailov, Zakhar Shumaylov, Yiren Zhao, Nicolas Papernot, Ross Anderson, and Yarin Gal. 2024. Ai models collapse when trained on recursively generated data. *Nature*, 631(8022):755–759.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*.

Hongjin Su, Shuyang Jiang, Yuhang Lai, Haoyuan Wu, Boao Shi, Che Liu, Qian Liu, and Tao Yu. 2024. Arks: Active retrieval in knowledge soup for code generation. *arXiv preprint arXiv:2402.12317*.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision.

Zhiqing Sun, Longhui Yu, Yikang Shen, Weiyang Liu, Yiming Yang, Sean Welleck, and Chuang Gan. 2024. Easy-to-hard generalization: Scalable alignment beyond human supervision.

Shuo Tang, Xianghe Pang, Zexi Liu, Bohan Tang, Rui Ye, Xiaowen Dong, Yanfeng Wang, and Siheng Chen. 2024. Synthesizing post-training data for llms through multi-agent simulation.

Rohan Taori, Ishaan Gulrajani, Tianyi Zhang, Yann Dubois, Xuechen Li, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca.

Ross Taylor, Marcin Kardas, Guillem Cucurull, Thomas Scialom, Anthony Hartshorn, Elvis Saravia, Andrew Poulton, Viktor Kerkez, and Robert Stojnic. 2022. Galactica: A large language model for science.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of llms via imagination, searching, and criticizing.

Yijun Tian, Huan Song, Zichen Wang, Haozhu Wang, Ziqing Hu, Fang Wang, Nitesh V. Chawla, and Panpan Xu. 2023. Graph neural prompting with large language models.

Shubham Toshniwal, Wei Du, Ivan Moshkov, Branislav Kisacanin, Alexan Ayrapetyan, and Igor Gitman. 2024a. Openmathinstruct-2: Accelerating ai for math with massive open-source instruction data.

Shubham Toshniwal, Ivan Moshkov, Sean Narenthiran, Daria Gitman, Fei Jia, and Igor Gitman. 2024b. Openmathinstruct-1: A 1.8 million math instruction tuning dataset. *arXiv preprint arXiv:2402.10176*.

Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, et al. 2023. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*.

Tao Tu, Anil Palepu, Mike Schaekermann, Khaled Saab, Jan Freyberg, Ryutaro Tanno, Amy Wang, Brenna Li, Mohamed Amin, Nenad Tomasev, Shekoofeh Azizi, Karan Singhal, Yong Cheng, Le Hou, Albert Webson, Kavita Kulkarni, S Sara Mahdavi, Christopher Semturs, Juraj Gottweis, Joelle Barral, Katherine Chou, Greg S Corrado, Yossi Matias, Alan Karthikesalingam, and Vivek Natarajan. 2024. Towards conversational diagnostic ai.

Dennis Ulmer, Elman Mansimov, Kaixiang Lin, Justin Sun, Xibin Gao, and Yi Zhang. 2024. Bootstrapping llm-based task-oriented dialogue agents via self-talk.

Pablo Villalobos, Anson Ho, Jaime Sevilla, Tamay Besiroglu, Lennart Heim, and Marius Hobbhahn. Position: Will we run out of data? limits of llm scaling based on human-generated data. In *Forty-first International Conference on Machine Learning*.

Chaojie Wang, Yanchen Deng, Zhiyi Lyu, Liang Zeng, Jujie He, Shuicheng Yan, and Bo An. 2024a. Q*: Improving multi-step reasoning for llms with deliberative planning.

Peiyi Wang, Lei Li, Zhihong Shao, Runxin Xu, Damai Dai, Yifei Li, Deli Chen, Yu Wu, and Zhifang Sui. 2024b. Math-shepherd: Verify and reinforce llms step-by-step without human annotations. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 9426–9439.

Ruiyi Wang, Haofei Yu, Wenxin Zhang, Zhengyang Qi, Maarten Sap, Graham Neubig, Yonatan Bisk, and Hao Zhu. 2024c. Sotopia-π: Interactive learning of socially intelligent language agents.

Tianlu Wang, Ilia Kulikov, Olga Golovneva, Ping Yu, Weizhe Yuan, Jane Dwivedi-Yu, Richard Yuanzhe Pang, Maryam Fazel-Zarandi, Jason Weston, and Xian Li. 2024d. Self-taught evaluators.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023a. Self-consistency improves chain of thought reasoning in language models.

Yixu Wang, Yan Teng, Kexin Huang, Chengqi Lyu, Songyang Zhang, Wenwei Zhang, Xingjun Ma, and Yingchun Wang. 2023b. Fake alignment: Are llms really aligned well? *arXiv preprint arXiv:2311.05915*.

Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023c. Self-instruct: Aligning language models with self-generated instructions.

Zengzhi Wang, Rui Xia, and Pengfei Liu. 2023d. Generative ai for math: Part i–mathpile: A billion-token-scale pretraining corpus for math. *arXiv preprint arXiv:2312.17120*.

Zhichao Wang, Bin Bi, Can Huang, Shiva Kumar Pentyala, Zixu James Zhu, Sitaram Asur, and Na Claire Cheng. 2025. Una: Unifying alignments of rlhf/ppo, dpo and kto by a generalized implicit reward function.

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

Jason Wei, Yi Tay, Rishi Bommasani, Colin Raffel, Barret Zoph, Sebastian Borgeaud, Dani Yogatama, Maarten Bosma, Denny Zhou, Donald Metzler, et al. 2022b. Emergent abilities of large language models. *arXiv preprint arXiv:2206.07682*.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023a. Chain-of-thought prompting elicits reasoning in large language models.

Tianwen Wei, Liang Zhao, Lichang Zhang, Bo Zhu, Lijie Wang, Haihua Yang, Biye Li, Cheng Cheng, Weiwei Lü, Rui Hu, et al. 2023b. Skywork: A more open bilingual foundation model. *arXiv preprint arXiv:2310.19341*.

Yixuan Weng, Minjun Zhu, Fei Xia, Bin Li, Shizhu He, Shengping Liu, Bin Sun, Kang Liu, and Jun Zhao. 2023. Large language models are better reasoners with self-verification.

Ronald J Williams. 1987. *Reinforcement-learning connectionist systems*. College of Computer Science, Northeastern University.

Shengguang Wu, Keming Lu, Benfeng Xu, Junyang Lin, Qi Su, and Chang Zhou. 2023a. Self-evolved diverse data sampling for efficient instruction tuning.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024a. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge.

Yue Wu, Zhiqing Sun, Huizhuo Yuan, Kaixuan Ji, Yiming Yang, and Quanquan Gu. 2024b. Self-play preference optimization for language model alignment.

Zeqiu Wu, Yushi Hu, Weijia Shi, Nouha Dziri, Alane Suhr, Prithviraj Ammanabrolu, Noah A. Smith, Mari Ostendorf, and Hannaneh Hajishirzi. 2023b. Fine-grained human feedback gives better rewards for language model training.

Xiao Xia, Dan Zhang, Zibo Liao, Zhenyu Hou, Tianrui Sun, Jing Li, Ling Fu, and Yuxiao Dong. 2024. Scenegenagent: Precise industrial scene generation with coding agent. *arXiv preprint arXiv:2410.21909*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, and Daxin Jiang. 2023. Wizardlm: Empowering large language models to follow complex instructions.

Fangzhi Xu, Qiushi Sun, Kanzhi Cheng, Jun Liu, Yu Qiao, and Zhiyong Wu. 2024a. Interactive evolution: A neural-symbolic self-training framework for large language models.

Yifan Xu, Xiao Liu, Xinghan Liu, Zhenyu Hou, Yueyan Li, Xiaohan Zhang, Zihan Wang, Aohan Zeng, Zhengxiao Du, Wenyi Zhao, et al. 2024b. Chatglm-math: Improving math problem-solving in large language models with a self-critique pipeline. *arXiv preprint arXiv:2404.02893*.

Yifan Xu, Xiao Liu, Xueqiao Sun, Siyi Cheng, Hao Yu, Hanyu Lai, Shudan Zhang, Dan Zhang, Jie Tang, and Yuxiao Dong. 2024c. Androidlab: Training and systematic benchmarking of android autonomous agents.

Lilong Xue, Dan Zhang, Yuxiao Dong, and Jie Tang. 2024. Autore: Document-level relation extraction with large language models. *arXiv preprint arXiv:2403.14888*.

An Yang, Beichen Zhang, Binyuan Hui, Bofei Gao, Bowen Yu, Chengpeng Li, Dayiheng Liu, Jianhong Tu, Jingren Zhou, Junyang Lin, Keming Lu, Mingfeng Xue, Runji Lin, Tianyu Liu, Xingzhang Ren, and Zhenru Zhang. 2024a. Qwen2.5-math technical report: Toward mathematical expert model via self-improvement.

Chengrun Yang, Xuezhi Wang, Yifeng Lu, Hanxiao Liu, Quoc V. Le, Denny Zhou, and Xinyun Chen. 2024b. Large language models as optimizers.

Kevin Yang, Dan Klein, Asli Celikyilmaz, Nanyun Peng, and Yuandong Tian. 2024c. Rlcd: Reinforcement learning from contrastive distillation for language model alignment.

Shuo Yang, Wei-Lin Chiang, Lianmin Zheng, Joseph E Gonzalez, and Ion Stoica. 2023. Rethinking benchmark and contamination for language models with rephrased samples. *arXiv preprint arXiv:2311.04850*.

Shunyu Yao, Howard Chen, John Yang, and Karthik Narasimhan. 2022a. Webshop: Towards scalable real-world web interaction with grounded language agents. *Advances in Neural Information Processing Systems*, 35:20744–20757.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models.

Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2022b. React: Synergizing reasoning and acting in language models. *arXiv preprint arXiv:2210.03629*.

Sha Yuan, Hanyu Zhao, Zhengxiao Du, Ming Ding, Xiao Liu, Yukuo Cen, Xu Zou, Zhilin Yang, and Jie Tang. 2021. Wudaocorpora: A super large-scale chinese corpora for pre-training language models. *AI Open*, 2:65–68.

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

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023a. Scaling relationship on learning mathematical reasoning with large language models.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Chuanqi Tan, and Chang Zhou. 2023b. Scaling relationship on learning mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023c. Rrhf: Rank responses to align language models with human feedback without tears.

Zheng Yuan, Hongyi Yuan, Chuanqi Tan, Wei Wang, Songfang Huang, and Fei Huang. 2023d. Rrhf: Rank responses to align language models with human feedback without tears.

Xiang Yue, Xingwei Qu, Ge Zhang, Yao Fu, Wenhao Huang, Huan Sun, Yu Su, and Wenhu Chen. 2023. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*.

Xiang Yue, Tuney Zheng, Ge Zhang, and Wenhu Chen. 2024. Mammoth2: Scaling instructions from the web.

Eric Zelikman, Georges Harik, Yijia Shao, Varuna Jayasiri, Nick Haber, and Noah D. Goodman. 2024. Quiet-star: Language models can teach themselves to think before speaking.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning.

Rowan Zellers, Ari Holtzman, Yonatan Bisk, Ali Farhadi, and Yejin Choi. 2019. Hellaswag: Can a machine really finish your sentence?

Aohan Zeng, Xiao Liu, Zhengxiao Du, Zihan Wang, Hanyu Lai, Ming Ding, Zhuoyi Yang, Yifan Xu, Wendi Zheng, Xiao Xia, et al. 2022. Glm-130b: An open bilingual pre-trained model. *arXiv preprint arXiv:2210.02414*.

Dan Zhang, Ziniu Hu, Sining Zhoubian, Zhengxiao Du, Kaiyu Yang, Zihan Wang, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. Sciglm: Training scientific language models with self-reflective instruction annotation and tuning. *arXiv preprint arXiv:2401.07950*.

Dan Zhang, Sining Zhoubian, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024b. Rest-mcts*: Llm self-training via process reward guided tree search. *arXiv preprint arXiv:2406.03816*.

Fengji Zhang, Bei Chen, Yue Zhang, Jacky Keung, Jin Liu, Daoguang Zan, Yi Mao, Jian-Guang Lou, and Weizhu Chen. 2023a. Repocoder: Repository-level code completion through iterative retrieval and generation. *arXiv preprint arXiv:2303.12570*.

Jiwen Zhang, Jihao Wu, Yihua Teng, Minghui Liao, Nuo Xu, Xiao Xiao, Zhongyu Wei, and Duyu Tang. 2024c. Android in the zoo: Chain-of-action-thought for gui agents. *arXiv preprint arXiv:2403.02713*.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, et al. 2022. Opt: Open pre-trained transformer language models. *arXiv preprint arXiv:2205.01068*.

Wenqi Zhang, Yongliang Shen, Linjuan Wu, Qiuying Peng, Jun Wang, Yueting Zhuang, and Weiming Lu. 2024d. Self-contrast: Better reflection through inconsistent solving perspectives.

Yifan Zhang, Jingqin Yang, Yang Yuan, and Andrew Chi-Chih Yao. 2023b. Cumulative reasoning with large language models. *arXiv preprint arXiv:2308.04371*.

Yifan Zhang, Yang Yuan, and Andrew Chi-Chih Yao. 2024e. On the diagram of thought.

Yunxiang Zhang, Muhammad Khalifa, Lajanugen Logeswaran, Jaekyeom Kim, Moontae Lee, Honglak Lee, and Lu Wang. 2024f. Small language models need strong verifiers to self-correct reasoning. *arXiv preprint arXiv:2404.17140*.

Xinyu Zhao, Fangcong Yin, and Greg Durrett. 2024a. Understanding synthetic context extension via retrieval heads.

Xufeng Zhao, Mengdi Li, Wenhao Lu, Cornelius Weber, Jae Hee Lee, Kun Chu, and Stefan Wermter. 2024b. Enhancing zero-shot chain-of-thought reasoning in large language models through logic.

Chenyu Zheng, Guoqiang Wu, Fan Bao, Yue Cao, Chongxuan Li, and Jun Zhu. 2023. Revisiting discriminative vs. generative classifiers: Theory and implications. In *International Conference on Machine Learning*, pages 42420–42477. PMLR.

Tianyu Zheng, Shuyue Guo, Xingwei Qu, Jiawei Guo, Xinrun Du, Qi Jia, Chenghua Lin, Wenhao Huang, Jie Fu, and Ge Zhang. 2024. Kun: Answer polishment for chinese self-alignment with instruction back-translation.

Li Zhong, Zilong Wang, and Jingbo Shang. 2024. Ldb: A large language model debugger via verifying runtime execution step-by-step. *arXiv preprint arXiv:2402.16906*.

Aojun Zhou, Ke Wang, Zimu Lu, Weikang Shi, Sichun Luo, Zipeng Qin, Shaoqing Lu, Anya Jia, Linqi Song, Mingjie Zhan, and Hongsheng Li. 2023a. Solving challenging math word problems using gpt-4 code interpreter with code-based self-verification.

Denny Zhou, Nathanael Schärli, Le Hou, Jason Wei, Nathan Scales, Xuezhi Wang, Dale Schuurmans, Claire Cui, Olivier Bousquet, Quoc Le, and Ed Chi. 2023b. Least-to-most prompting enables complex reasoning in large language models.

Shuyan Zhou, Frank F. Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, Uri Alon, and Graham Neubig. 2024a. Webarena: A realistic web environment for building autonomous agents.

Shuyan Zhou, Frank F Xu, Hao Zhu, Xuhui Zhou, Robert Lo, Abishek Sridhar, Xianyi Cheng, Tianyue Ou, Yonatan Bisk, Daniel Fried, et al. 2023c. Webarena: A realistic web environment for building autonomous agents. *arXiv preprint arXiv:2307.13854*.

Yifei Zhou, Andrea Zanette, Jiayi Pan, Sergey Levine, and Aviral Kumar. 2024b. Archer: Training language model agents via hierarchical multi-turn rl. *arXiv preprint arXiv:2402.19446*.

Yongchao Zhou, Andrei Ioan Muresanu, Ziwen Han, Keiran Paster, Silviu Pitis, Harris Chan, and Jimmy Ba. 2022. Large language models are human-level prompt engineers. *arXiv preprint arXiv:2211.01910*.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2023d. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning.

Yuqi Zhu, Jia Li, Ge Li, YunFei Zhao, Jia Li, Zhi Jin, and Hong Mei. 2023. Hot or cold? adaptive temperature sampling for code generation with large language models.

A Useful Blog Posts and Code Repositories

In addition to the works mentioned in the body, the following blog posts and code repositories may be helpful.

- OpenAI o1 System Card: https://cdn.openai.com/o1-system-card-20241205.pdf
- Learning to Reason with LLMs: https://openai.com/index/learning-to-reason-with-llms/
- OpenAI's Strawberry, LM self-talk, inference scaling laws, and spending more on inference: https://www.interconnects.ai/p/openai-strawberry-and-inference-scaling-laws
- Improving LLM Reasoning using Self-generated Data: https://drive.google.com/file/d/1komQ7s9kPPvDx_8AxTh9A6tlfJA0j6dR/edit
- QwQ: Reflect Deeply on the Boundaries of the Unknown: https://qwenlm.github.io/blog/qwq-32b-preview
- DeepSeek-R1-Lite-Preview: https://api-docs.deepseek.com/news/news1120
- Ilya Sutskever's talk at NeurIPS 2024: https://x.com/vincentweisser/status/1867719020444889118

B Additional Post-Training Works

See Table 1 for additional SFT works, Table 2 for RL works, and Table 3 for TTC works. Each work is briefly described.

Paper/Work	Brief Description
Guan et al. (2024)	The paper proposes verifier engineering, a novel post-training approach for enhancing LLMs. It uses automated verifiers to perform verification and provide feedback. The process is divided into three stages: search, verify, and feedback.
Tang et al. (2024)	MATRIX is a multi-agent simulator designed to create realistic and scalable text-based scenarios by simulating interactions in human society. This method enables effective post-training of LLMs, producing both general and domain-specific data.
Qu et al. (2024)	Recursive IntroSpEction (RISE) is a method for fine-tuning LLMs to enable them to iteratively improve their responses by introspecting and correcting mistakes over multiple turns. RISE fine-tunes models using an iterative procedure, treating fine-tuning for a single-turn prompt as solving a multi-turn Markov decision process.
Zhao et al. (2024a)	This paper explores using synthetic data to fine-tune LLMs' handling of retrieval and reasoning in long-context tasks. The research varies the realism of key "needle" concepts and the diversity of surrounding "haystack" contexts, comparing models trained on synthetic data versus real data.
Luo et al. (2024a)	This paper introduces Arena Learning, an offline strategy for evaluating and enhancing LLMs by simulating human-annotated battles typically conducted in online Chatbot Arenas. Arena Learning employs AI-driven annotations to simulate battle outcomes, enabling continuous model improvement through SFT.

Table 1: Additional post-training works for SFT.

Paper/Work	Brief Description
Kumar et al. (2024)	SCoRe is a multi-turn online reinforcement learning approach designed to enhance the self-correction ability of LLMs using entirely self-generated data. SCoRe addresses the challenges of distribution mismatch and behavior collapse inherent in supervised fine-tuning.
Bukharin et al. (2024)	HERON is a hierarchical reward modeling framework that simplifies the reward design process in reinforcement learning. It uses a hierarchical decision tree based on the importance ranking of feedback signals to compare trajectories and train a reward model for policy learning.
Wu et al. (2024b)	Self-Play Preference Optimization (SPPO) is a novel method for language model alignment that frames the problem as a constant-sum two-player game aimed at identifying the Nash equilibrium policy.
Lambert et al. (2024a)	This paper introduces a new training method called Reinforcement Learning with Verifiable Rewards (RLVR). It trains LLMs on tasks with verifiable outcomes by replacing the reward model in RLHF with a verification function.
Chen et al. (2024a)	IterAlign is for aligning LLMs with human values without requiring extensive human annotations or predefined rules. IterAlign utilizes a process of red teaming to identify weaknesses in the LLM and leverages a stronger LLM to discover new constitutions for guiding the self-correction of the base model.

Table 2: Additional post-training works for RL.

Paper/Work	Brief Description
Ding et al. (2024)	Everything of Thoughts (XoT) is a novel thought-prompting approach that enhances LLMs by integrating RL and MCTS to incorporate external domain knowledge. XoT enhances LLMs' performance and efficiency by autonomously creating quality cognitive mappings with minimal input, enabling flexible problem-solving with multiple solutions.
Feng et al. (2024)	The paper proposes an AlphaZero-like tree-search learning framework called TS-LLM, which integrates a learned value function with tree-search algorithms for guiding LLM decoding.
Liu et al. (2024a)	PPO-MCTS is a novel value-guided decoding algorithm that integrates MCTS with the PPO value network for enhanced natural language text generation. This approach utilizes the value network, a byproduct of PPO training, to evaluate partial output sequences during inference, thereby aligning the scoring mechanisms between the training and testing phases.
Tian et al. (2023)	The paper proposes Graph Neural Prompting (GNP) as a new method to integrate grounded knowledge from knowledge graphs into LLMs. GNP is designed to be a plug-and-play solution, includes a standard graph neural network encoder, a cross-modality pooling module, and a domain projector, and employs a self-supervised link prediction objective.
Liang et al. (2024)	This paper proposes scaling up inference-time computation by generating multiple reasoning paths and using verifiers to assess and rank outputs based on correctness. It integrates Chain-of-Thought (CoT) and Program-of-Thought (PoT) for collaborative verification.

Table 3: Additional post-training works for TTC.