Search, Verify and Feedback: Towards Next Generation Post-training Paradigm of Foundation Models via Verifier Engineering

Xinyan Guan[*] **Yanjiang Liu**[*] **Xinyu Lu**[*] **Boxi Cao** **Ben He** **Xianpei Han** **Le Sun**
Jie Lou[†] **Bowen Yu**[†] **Yaojie Lu**[†] **Hongyu Lin**[†]

Chinese Information Processing Laboratory,
Institute of Software, Chinese Academy of Sciences
E-mail to: hongyu@iscas.ac.cn
https://github.com/icip-cas/Verifier-Engineering

Abstract

The evolution of machine learning has increasingly prioritized the development of powerful models and more scalable supervision signals. However, the emergence of foundation models presents significant challenges in providing effective supervision signals necessary for further enhancing their capabilities. Consequently, there is an urgent need to explore novel supervision signals and technical approaches. In this paper, we propose ***verifier engineering***, a novel post-training paradigm specifically designed for the era of foundation models. The core of verifier engineering involves leveraging a suite of automated verifiers to perform verification tasks and deliver meaningful feedback to foundation models. We systematically categorize the verifier engineering process into three essential stages: search, verify, and feedback, and provide a comprehensive review of state-of-the-art research developments within each stage. We believe that verifier engineering constitutes a fundamental pathway toward achieving Artificial General Intelligence.

	Feature Engineering	Data Engineering	Verifier Engineering
Representative Models	Machine Learning Models	Deep Learning Models	Foundation Models
e.g.	SVM, XGBoost	CNN, LSTM	LLMs, VLMs
Supervision	Manual Features	Human Annotations	Verifier Feedback
Scope	Task-Specific	Multiple Related Tasks	General Intelligence
Generalization	Limited	Relatively high	High
Scalability	Limited	Moderate	High

Table 1: Comparison of feature engineering, data engineering, and verifier engineering

1. Introduction

The evolution of machine learning (Jordan & Mitchell, 2015) has undergone a progressive journey characterized by the pursuit of increasingly powerful models and the scaling of supervision signals (Friedland & Krell, 2018; Cao et al., 2024; Sevilla et al., 2022). Over the past decades, both model capacity and the scale of supervision signals have escalated in a synergistic manner. Powerful models necessitate more scalable and effective supervision signals to fully utilize their parameters. Conversely, the expansion of supervision signals requires models with enhanced capacities to effectively exploit these signals, thereby achieving more generalized capabilities.

In the early stages of machine learning, models were constrained by their limited capacity. This period was characterized as ***feature engineering***, during which domain experts manually designed and extracted relevant features. Classical algorithms, such as Support Vector Machines (Hearst et al., 1998) and Decision Trees (Quinlan, 1986), relied heavily on meticulously crafted features due to their architectural limitations. These algorithms achieved optimal performance through carefully designed feature extraction techniques, with Term Frequency-Inverse Document Frequency (Robertson et al., 2009) serving as a prominent example.

However, as addressed increasingly complex problems, the limitations of manual feature engineering became more pronounced, underscoring the need for more scalable approaches to construct features. The emergence of deep learning (Schmidhuber, 2015) approximately two decades ago marked a transformative shift, inaugurating the ***data engineering*** era. This new paradigm represented a funda-

[*]Equal contribution.
[†]Corresponding authors. Bowen Yu is affiliated with Qwen Team, Alibaba Group. Jie Lou is affiliated with Xiaohongshu Inc. This work is initiated by 4 corresponding authors in a personal meeting.

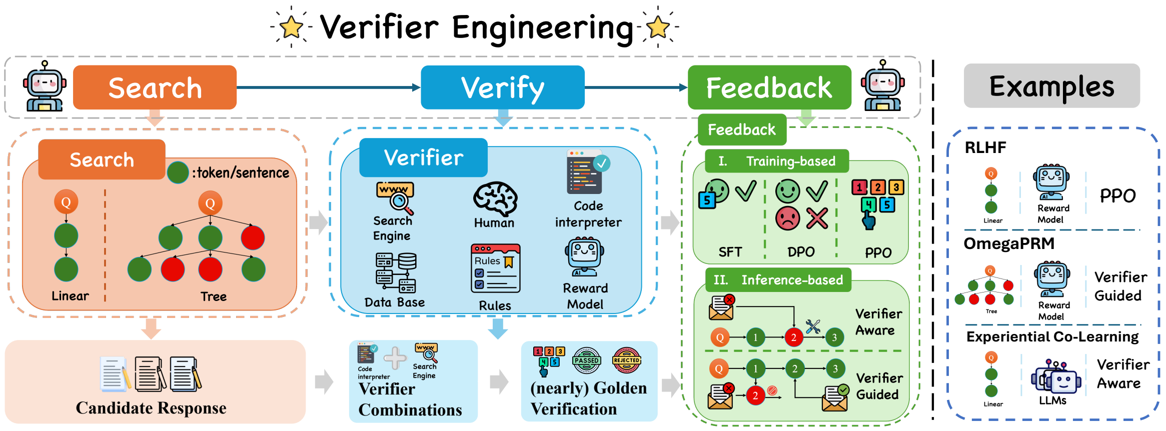

Figure 1: **Framework of *verifier engineering***: The fundamental stages of verifier engineering include Search, Verify, and Feedback. Given an instruction, the process begins with generating candidate responses (Search), followed by evaluating these candidates using appropriate verifier combinations (Verify), and concludes with optimizing the model's output distribution (Feedback). This framework can explain various approaches, from training-based methods like RLHF (Ouyang et al., 2022a) to inference-based techniques such as OmegaPRM (Luo et al., 2024b) and Experiential Co-Learning (Qian et al., 2023). We systematically categorize existing approaches into these three stages in Table 3.

mental departure from handcrafted features, emphasizing instead the curation of high-quality datasets and annotations to facilitate automated knowledge acquisition and pattern recognition across diverse domains and tasks. The remarkable success of landmark projects such as ImageNet (Deng et al., 2009) and BERT (Brown et al., 2020a) validated the effectiveness of this data-centric approach.

Unfortunately, the emergence of foundation models in recent years (Ouyang et al., 2022a; Touvron et al., 2023a; Betker et al.; Dosovitskiy et al., 2021) has made it increasingly difficult to enhance model capabilities solely through data engineering. Specifically, foundation models, particularly large language models (LLMs) (Ouyang et al., 2022a; Touvron et al., 2023a), have demonstrated extraordinary abilities, increasingly matching or surpassing human performance across various domains. Nevertheless, the traditional data engineering approach of augmenting these models through large-scale data construction has reached its practical limitations. This limitation is evident in two primary challenges: the difficulty and unsustainable costs associated with high-quality human annotations for post-training (Anthropic, 2024; Burns et al., 2023), and the complexity involved in providing meaningful guidance that can further enhance model performance (Wen et al., 2024a). Consequently, the central challenge in the current era is determining how to supply more effective supervision signals to foundation models to achieve general artificial intelligence capabilities.

In this paper, we propose ***verifier engineering***, a novel post-training paradigm designed for the foundation model era. The essence of verifier engineering lies in extending the construction of supervision signals beyond traditional manual feature extraction and data annotation. Instead, it utilizes a suite of effective automated verifiers to perform verification tasks and provide meaningful feedback to foundation models. Table 1 delineates the key distinctions among feature engineering, data engineering, and verifier engineering. This progression from **annotate and learn** to **search and verify** signifies a fundamental advancement in enhancing the capabilities of foundation models. Compared to preceding paradigms, verifier engineering streamlines the creation of verifiers and facilitates efficient feedback to foundation models through automated verification processes. Specifically, given an instruction, verifier engineering initiates by generating candidate responses, subsequently verifying these candidates using appropriate combinations of verifiers, and ultimately optimizes the model's output distribution. Unlike existing methodologies such as Reinforcement Learning from Human Feedback (RLHF) (Stiennon et al., 2020; Ouyang et al., 2022a), which depend on a limited set of verifier sources and feedback mechanisms, verifier engineering integrates multiple diverse verifiers to deliver more accurate and generalizable feedback signals. By shifting the improvement of foundation models from a data-centric endeavor to a systematic engineering challenge, verifier engineering emphasizes the design and orchestration of complex verification systems to ensure effective and efficient feedback. Analogous to how feature engineering and data engineering achieved scalability in their respective eras, we posit that verifier engineering represents a crucial step toward the advancement of general artificial intelligence.

To this end, this paper provides a comprehensive exploration

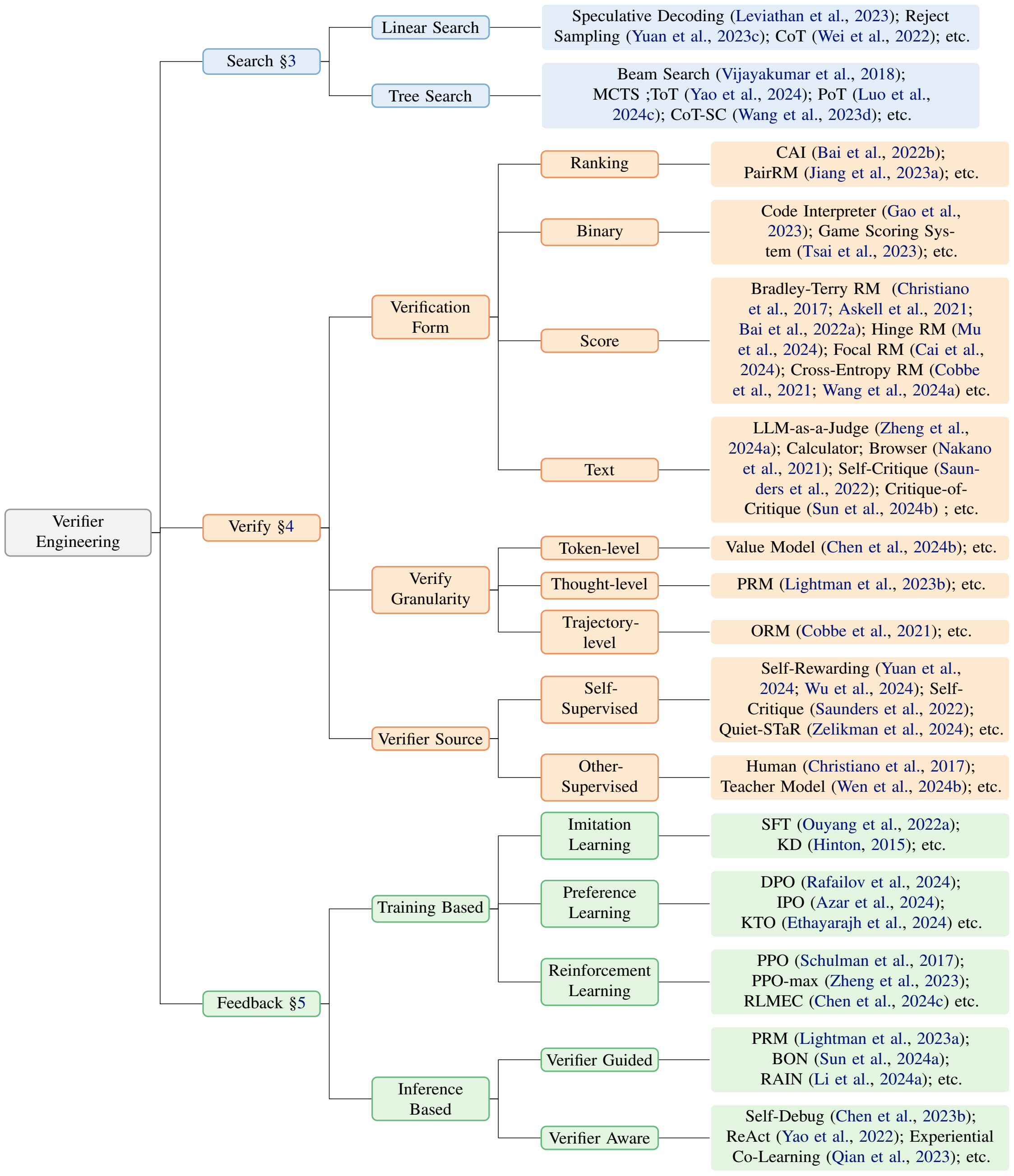
Verifier Engineering
Search §3
Linear Search
Speculative Decoding (Leviathan et al., 2023); Reject Sampling (Yuan et al., 2023c); CoT (Wei et al., 2022); etc.
Tree Search
Beam Search (Vijayakumar et al., 2018); MCTS ;ToT (Yao et al., 2024); PoT (Luo et al., 2024c); CoT-SC (Wang et al., 2023d); etc.
Verify §4
Verification Form
Ranking
CAI (Bai et al., 2022b); PairRM (Jiang et al., 2023a); etc.
Binary
Code Interpreter (Gao et al., 2023); Game Scoring System (Tsai et al., 2023); etc.
Score
Bradley-Terry RM (Christiano et al., 2017; Askell et al., 2021; Bai et al., 2022a); Hinge RM (Mu et al., 2024); Focal RM (Cai et al., 2024); Cross-Entropy RM (Cobbe et al., 2021; Wang et al., 2024a) etc.
Text
LLM-as-a-Judge (Zheng et al., 2024a); Calculator; Browser (Nakano et al., 2021); Self-Critique (Saunders et al., 2022); Critique-of-Critique (Sun et al., 2024b) ; etc.
Verify Granularity
Token-level
Value Model (Chen et al., 2024b); etc.
Thought-level
PRM (Lightman et al., 2023b); etc.
Trajectory-level
ORM (Cobbe et al., 2021); etc.
Verifier Source
Self-Supervised
Self-Rewarding (Yuan et al., 2024; Wu et al., 2024); Self-Critique (Saunders et al., 2022); Quiet-STaR (Zelikman et al., 2024); etc.
Other-Supervised
Human (Christiano et al., 2017); Teacher Model (Wen et al., 2024b); etc.
Feedback §5
Training Based
Imitation Learning
SFT (Ouyang et al., 2022a); KD (Hinton, 2015); etc.
Preference Learning
DPO (Rafailov et al., 2024); IPO (Azar et al., 2024); KTO (Ethayarajh et al., 2024) etc.
Reinforcement Learning
PPO (Schulman et al., 2017); PPO-max (Zheng et al., 2023); RLMEC (Chen et al., 2024c) etc.
Inference Based
Verifier Guided
PRM (Lightman et al., 2023a); BON (Sun et al., 2024a); RAIN (Li et al., 2024a); etc.
Verifier Aware
Self-Debug (Chen et al., 2023b); ReAct (Yao et al., 2022); Experiential Co-Learning (Qian et al., 2023); etc.

Figure 2: Overview of verifier engineering methodologies, categorized into three main stages: Search, Verify, and Feedback. Each stage is further broken down into specific approaches, with references to notable works in each area.

of the verifier engineering landscape, breaking it down into three core stages: search, verify, and feedback. These stages are defined as follows:

1. **Search**: Sampling representative or potentially problematic samples from model output distribution to reveal performance boundaries and limitations.
2. **Verify**: Utilizing various verifiers to provide verification results on candidate responses, which may include evaluation metrics, rule detection, or selective manual annotation.
3. **Feedback**: Leveraging verification results to enhance model performance through methods such as Supervised Fine-Tuning or In-Context Learning.

To showcase the latest state-of-the-art research developments in these three stages, we provide a comprehensive review in the remainder of this paper: Section 2 presents the formal definition and preliminaries of verifier engineering. Sections 3-5 detail the three fundamental stages: search, verify, and feedback. Section 6 discusses current trends and limitations, followed by concluding remarks in Section 7.

2. Verifier Engineering

In this section, we formalize verifier engineering as a Goal-Conditioned Markov Decision Process (GC-MDP) (Schaul et al., 2015; Plappert et al., 2018; Liu et al., 2022), enabling a unified and systematic perspective on the field of verifier engineering. Then, we introduce how the concepts of search, verify, and feedback correspond within this modeling framework, and analyze them through examples. Furthermore, we provide a comprehensive overview of the verifier engineering landscape by categorizing existing post-training approaches into three core stages, as summarized in Table 3.

2.1. Preliminary

While LLMs are typically trained to maximize generation likelihood given input, this objective alone cannot guarantee desired post-training capabilities. To bridge this gap, we formalize verifier engineering as a GC-MDP, denoted as tuple (S, A, T, G, R_g, p_g), where:

State Space S represents the model's state during interaction, including input context, internal states, and intermediate outputs.

Action Space A represents the possible token selections at each generation step:

$$A = \{a_1, a_2, \ldots, a_N\}$$

Transition Function T typically defines the probability distribution over the next states, given the current state $s \in S$ and action $a \in A$. Specifically, in large language models, the state transition is a deterministic function. That is, once the current state and selected action (the generated token) are given, the next state is fully determined. The search stage of the verifier engineering thus can be regarded as exploration in the action-state space in the condition of T.

Goal Space G represents the various goals related to model capabilities. Each goal $g \in G$ corresponds to a specific model capability, such as code, math, writing, and so on. The goal space is multi-dimensional and can encompass various aspects of model capabilities.

Goal Distribution p_g is the probability distribution over goals from the goal space G. It represents the likelihood of a specific goal $g \in G$ being selected at any given time. This distribution can be learned from human feedback or other external signals.

Reward Function $R_g(s, a)$ **or** $R_g(s')$ represents the reward the model receives given the goal g, when taking action a from state s or at the transformed state s'. This reward thus reflects the verification result of verifier engineering for the specific goal g. For example, if the goal g is ”fairness,” the reward might be based on whether the generated text avoids bias.

The objective for improving model capabilities can be defined as a goal-conditioned policy $\pi : S \times G \times A \to [0, 1]$ that maximizes the expectation of the cumulative return over the goal distribution.

$$J(\pi) = \mathbb{E}_{\substack{a_t \sim \pi(\cdot|s_t, g), g \sim p_g \\ s_{t+1} \sim \mathcal{T}(\cdot|s_t, a_t)}} \left[\sum_t \gamma^t R_g(s_t, a_t) \right] \quad (1)$$

The reward function can be decomposed into sub-functions for different capability dimensions:

$$R_g(s, a) = \mathcal{F}\left(R_{g,i}(s, a) \mid i \in \mathcal{S}_g\right) \quad (2)$$

where $\mathcal{S}_g$ represents selected sub-functions for goal g, and $\mathcal{F}$ combines their evaluations. Following Probably Approximately Correct Theory (PAC) (Vapnik, 2000), even with imperfect sub-functions, we can achieve reliable overall evaluation by combining multiple weak verifiers.

2.2. Verifier Engineering Overview

Building upon the GC-MDP framework, we demonstrate how the three stages of verifier engineering—**Search**, **Verify**, and **Feedback**—naturally align with specific components of this formalism. This mapping provides both theoretical grounding and insights into how each stage contributes to optimizing the policy distribution π toward desired goals.

As illustrated in Figure 1, the search stage can be divided into linear search and tree search methods. The verify stage selects appropriate verifiers and invokes them to provide verification results based on the candidate responses. The feedback stage improve the model's output using either training-based methods or inference-based methods. For example, RLHF utilizes linear search to generate batches of responses, employs a reward model as the verifier, and applies the Proximal Policy Optimization (PPO) (Schulman et al., 2017) algorithm to optimize the model based on the reward model's verification result. OmegaPRM (Luo et al., 2024a) employs a process reward model as a verifier and searches for the best result based on the PRM by maximizing the process reward scores. Experimental Co-Learning (Qian et al., 2023) collaborative verifiers through multiple LLMs, enhancing both verify and model performance through historical interaction data.

In the GC-MDP framework, the three stages manifest as follows: Search corresponds to action selection, where state s represents the current context or partial sequence, and action a denotes the next token$s$ to generate. Verify maps to the reward function $R_g(s, a)$, evaluating how well the generated outputs align with the specified goal g. Feedback relates to policy distribution π optimization, maximizing the expected cumulative return $J(\pi)$ based on verification results.

3. Search

Search aims to identify high-quality generated sequences that align with the intended goals, forming the foundation of verifier engineering, which is critical for evaluating and refining foundation models. However, it is impractical to exhaustively search the entire state-action space due to its exponential growth—driven by the large vocabulary size N and the maximum generation length T. To address this challenge, efficient searching, which seeks to navigate this vast space by prioritizing diverse and goal-oriented exploration, plays a critical role in improving model's performance.

In this section, we first introduce how to implement diverse search from the perspective of search structure and discuss additional methods for further enhancing search diversity after the search structure is determined.

3.1. Search Structure

Search structure denotes the framework or strategy used to navigate the state-action space, which significantly influences the effectiveness and efficiency of the search process. Currently, there are two widely adopted structures for implementing search: linear search and tree search. Linear search progresses sequentially, making it effective for tasks involving step-by-step actions, while tree search examines multiple paths at each decision point, making it well-suited for tasks requiring complex reasoning.

- **Linear Search** is a widely used search method where the model starts from an initial state and proceeds step by step, selecting one token at a time until reaching the terminal state (Brown et al., 2020b; Wang & Zhou, 2024).

  The key advantage of linear search is its low computational cost, which makes it efficient in scenarios where actions can be selected sequentially to progress toward a goal. However, a notable limitation is that if a sub-optimal action is chosen early in the process, it may be challenging to rectify the subsequent sequence. Thus, during the progress of verifier engineering, careful verification at each step is crucial to ensure the overall effectiveness of the generation path.

- **Tree Search** involves exploring multiple potential actions at each step of generation, allowing for a broader exploration of the state-action space. For instance, techniques like Beam Search and ToT (Yao et al., 2024) incorporate tree structures to enhance exploration, improving model performance in reasoning tasks. TouT (Mo & Xin, 2023) introduces uncertainty measurement based on Monte Carlo dropout, providing a more accurate evaluation of intermediate reasoning processes.

  This approach significantly improves the likelihood of discovering global optimal solution, particularly in environments with complex state spaces. By simultaneously considering multiple paths, tree search mitigates the risk of being locked into sub-optimal decisions made early on, making it more robust in guiding the model toward an optimal outcome. However, this increased exploration comes at a higher computational cost, making tree search more suitable for scenarios when the optimal path is challenging to identify. To make tree search effective, the model must be continuously verified and prioritize paths that align better with the goal conditions.

3.2. Additional Enhancement

While search structures provide the foundational framework for navigating the state-action space, further enhancement is also crucial to enhance search performance. These enhancements address challenges such as balancing exploration and exploitation, escaping local optima, and improving the diversity of generated results. The enhancement strategies can be broadly categorized into two approaches: adjusting exploration parameters and intervening in the original state.

Adjusting Exploration Parameters Techniques such as Monte Carlo Tree Search (MCTS), Beam Search, and Reject Sampling focus on refining the exploration process by adjusting parameters like temperature, Top-k (Fan et al., 2018), or Top-p (Holtzman et al., 2020). The challenge lies in balancing the trade-off between generating diverse outputs and maintaining high-quality sequences. For example, increasing the temperature parameter promotes greater randomness, fostering diversity but potentially reducing coherence.

Intervening in the Original State Another enhancement approach involves modifying the initial state to guide the search process toward specific goals. Methods like Chain of Thought (CoT) (Wei et al., 2022), Logical Inference via Neurosymbolic Computation (LINC) (Olausson et al., 2023), and Program of Thoughts (PoT) (Chen et al., 2023a) exemplify this strategy. These interventions address the challenge of overcoming biases in the default state distribution. CoT enhances reasoning by introducing intermediate steps, improving interpretability and depth in generated sequences. LINC uses logical scenarios to encourage more structured and goal-oriented outputs. Similarly, PoT provides programmatic examples that guide models toward systematic problem-solving, effectively expanding the scope of exploration beyond the original state distribution.

4. Verify

Due to the long delay and high cost associated with human feedback, we cannot directly employ human efforts to evaluate each candidate response sampled by the model during training (Leike et al., 2018a). Therefore, we employ verifiers as proxies for human supervision in the training of foundation models. The verifiers play a crucial role in search-verify-feedback pipeline, and the quality and robustness of the verifiers directly impact the performance of the downstream policy (Wen et al., 2024c).

In the context of GC-MDP, verify is typically defined as using a verifier that provides verification results based on the current state and a predefined goal:

$$F \leftarrow R_g(s_{t-1}, a_t) \tag{3}$$

where F represents the verification results provided by the verifier. g denotes the predefined goal we aim to achieve (e.g., helpfulness, honesty). The state s is typically composed of two concatenated components: the user's query or input and the model's output content $\{a_1, \ldots, a_t\}$.

In this section, we classify individual verifiers across several key dimensions and summarize the representative type of verifiers in Table 2.

4.1. A Comprehensive Taxonomy of Verifiers

Perspective of Verification Form The verification result form of verifiers can be divided into four categories: binary feedback (Gao et al., 2023), score feedback (Bai et al., 2022a), ranking feedback (Jiang et al., 2023a), and text feedback (Saunders et al., 2022). These categories represent an increasing gradient of information richness, providing more information to the optimization algorithm. For instance, classic Bradley–Terry reward models (Bradley & Terry, 1952) can provide continuous score feedback which simply indicates correctness, while text-based feedback from generative reward models (Zhang et al., 2024d) and critique models (Sun et al., 2024b) offer more detailed information, potentially including rationals for scores or critiques.

Perspective of Verify Granularity The verify granularity of verifiers can be divided into three levels: token-level (Chen et al., 2024b), thought-level (Lightman et al., 2023a), and trajectory-level (Cobbe et al., 2021). These levels correspond to the scope at which the verifier engages with the model's generation. Token-level verifiers focus on individual next-token predictions, thought-level verifiers assess entire thought steps or sentences, and trajectory-level verifiers evaluate the overall sequence of actions. Although most RLHF practices (Ouyang et al., 2022b; Bai et al., 2022a) currently rely on full-trajectory scoring, coarse-grained ratings are challenging to obtain accurately (Wen et al., 2024b), as they involve aggregating finer-grained verification. Generally, from the perspective of human annotators, assigning finer-grained scores is easier when the full trajectory is visible. From a machine learning perspective, fine-grained verification is preferable (Lightman et al., 2023a), as it mitigates the risks of shortcut learning and bias associated with coarse-grained verification, thereby enhancing generalization. A credit-assignment mechanism (Leike et al., 2018b) can bridge the gap between coarse-grained ratings and fine-grained verification.

Perspective of Verifier Source Verifiers can be divided into program-based and model-based from the perspective of verifier source. Program-based verifiers provide deterministic verification, typically generated by predefined rules or logic embedded in fixed programs. These program-based verifiers offer consistent and interpretable evaluations but may lack flexibility when dealing with complex, dynamic environments. On the other hand, model-based verifiers rely on probabilistic models to generate verification results. These verifiers adapt to varying contexts and tasks through learning, allowing for more nuanced and context-sensitive evaluations. However, model-based verifiers introduce an element of uncertainty and can require significant training and computational resources to ensure accuracy and robustness.

Perspective of Extra Training Verifiers can also be divided into two categories based on whether they require additional specialized training. Verifiers requiring additional training are typically fine-tuned on specific task-related data, allowing them to achieve higher accuracy in particular problem domains (Markov et al., 2023). However, their performance can be heavily influenced by the distribution of the

Verifier Type	Verification Form	Verify Granularity	Verifier Source	Extra Training
Golden Annotation	Binary/Text	Thought Step/Full Trajectory	Program Based	No
Rule-based	Binary/Text	Thought Step/Full Trajectory	Program Based	No
Code Interpreter	Binary/Score/Text	Token/Thought Step/Full Trajectory	Program Based	No
ORM	Binary/Score/Rank/Text	Full Trajectory	Model Based	Yes
Language Model	Binary/Score/Rank/Text	Thought Step/Full Trajectory	Model Based	Yes
Tool	Binary/Score/Rank/Text	Token/Thought Step/Full Trajectory	Program Based	No
Search Engine	Text	Thought Step/Full Trajectory	Program Based	No
PRM	Score	Token/Thought Step	Model Based	Yes
Knowledge Graph	Text	Thought Step/Full Trajectory	Program Based	No

Table 2: **A comprehensive taxonomy of verifiers across four dimensions:** verification form, verify granularity, verifier source, and the need for extra training.

training data, potentially making them less generalizable to different contexts. On the other hand, verifiers that do not require extra training are often based on pre-existing models (Zelikman et al., 2024; Zheng et al., 2024a). While they may not reach the same level of accuracy as their task-specific counterparts, they are generally more robust to variations in data distribution, making them less dependent on specific training sets. This trade-off between accuracy and data sensitivity is a key consideration when selecting a verifier for a given application.

4.2. Constructing Verifiers across Tasks

Different tasks have varying requirements for verifiers, serving as critical tools to enhance the performance of foundation models. In this section, we will highlight key practices for constructing verifiers in representative fields, including safety and reasoning.

Safety In safety-critical applications, verifiers play a crucial role in ensuring that foundation models adhere to ethical standards and avoid generating harmful or inappropriate content. Program-based verifiers can enforce strict guidelines by filtering out outputs that include prohibited content, such as hate speech or sensitive personal information. For instance, a content moderation system might employ predefined keywords and patterns to identify and block offensive language. However, a limitation of program-based approaches is their vulnerability to adversarial attacks, as paraphrased content can often bypass these filters (Krishna et al., 2020). In contrast, model-based verifiers, such as toxicity classifiers (Lees et al., 2022; Markov et al., 2023; Inan et al., 2023), offer probabilistic assessments of content safety, enabling more nuanced evaluations. A middle-ground approach is rule-based reward models (RRMs) (Mu et al., 2024), which balance interpretability with generalization capabilities. Integrating verifiers into both the training and deployment phases allows foundation models to align more closely with safety requirements, reducing the likelihood of unintended or harmful outputs.

Reasoning In tasks requiring logical deduction or problem-solving, verifiers can assess the correctness and coherence of each reasoning step. Token-level verifiers offer fine-grained evaluations of individual tokens or symbols, which is particularly useful in mathematical computations and code generation (Wang et al., 2023b). Thought-level verifiers, on the other hand, evaluate entire sentences or reasoning steps to ensure that each component of the argument is both valid and logically consistent (Lightman et al., 2023a; Li et al., 2023b; Xie et al., 2024). Trajectory-level verifiers assess the overall solution or proof, providing comprehensive verification results on the coherence of the model's reasoning (Cobbe et al., 2021; Yu et al., 2024; Wang et al., 2024a). For instance, in mathematical theorem proving, a program-based verifier like Lean (de Moura et al., 2015) can check each proof step's validity against formal logic rules (Lin et al., 2024), while a model-based verifier can assess the plausibility of reasoning steps through scores and natural language explanations (Zhang et al., 2024c), offering critiques for further refinement (Kumar et al., 2024). A simpler yet widely-used approach involves using manually annotated correct answers as verifiers to filter model outputs, progressively enhancing model performance (Zelikman et al., 2022b).

5. Feedback

After obtaining the verification result, we aim to enhance the capabilities of the foundation model, we define this process as the feedback stage. In this paper, feedback specifically refers to enhancing the foundation model's capabilities based on the verification results. The feedback stage is critical, as the effectiveness of the feedback method directly determines whether the foundation model's capabilities can be appropriately enhanced in response to the verification results.

In this section, we explore how verifier engineering utilizes search algorithms and verifiers to feedback verification results on foundation models. To maximize the objective function $J(\pi)$, the distribution of the policy π can be optimized by adjusting s_t or the parameters of π. This leads to two distinct feedback approaches: training-based feedback and inference-based feedback. Training-based feedback involves updating model parameters using data efficiently obtained through searching and verifying. In contrast, inference-based feedback modifies the output distribution by incorporating search and verification results as auxiliary information during inference, without altering the model parameters.

5.1. Training-based Feedback

We categorize the common training strategies for training-based feedback into three types, based on the nature and organization of the data utilized:

Imitation Learning Imitation Learning typically uses high-quality data selected by verifiers, employing supervised fine-tuning objectives like cross-entropy or knowledge distillation training objectives like Kullback-Leibler divergence (Hinton, 2015) to optimize the model.

Various approaches are employed to enhance specific capabilities of the foundation model through imitation learning. To enhance mathematical reasoning capabilities in LLMs, approaches like STaR (Zelikman et al., 2022b) and RFT (Yuan et al., 2023b) use rule-based verifiers to compare solution outcomes, while WizardMath (Luo et al., 2023a), MARIO (Liao et al., 2024), and MetaMath (Yu et al., 2023a) leverage verifiers to select responses from advanced LLMs or human inputs. Other methods such as MAmmoTH (Yue et al., 2023) and MathCoder (Wang et al., 2023a) utilize programmatic tools to verify comprehensive and detailed solutions. For coding capability improvement, various methods including Code Alpaca (Chaudhary, 2023), WizardCoder (Luo et al., 2023b), WaveCoder (Yu et al., 2023b), and OpenCodeInterpreter (Zheng et al., 2024b) construct code instruction-following datasets by distilling knowledge from advanced LLMs. To enhance instruction-following abilities, approaches like LLaMA-GPT4 (Peng et al., 2023), Baize (Xu et al., 2023), and Ultrachat (Ding et al., 2023) employ verifiers to distill responses from advanced LLMs for supervised fine-tuning. Other methods such as Deita (Liu et al., 2024) and MoDS (Du et al., 2023) implement a pipeline of verifiers to check complexity, quality, and diversity before selecting suitable data for SFT.

Preference Learning Preference Learning leverages verification results to construct pairwise comparison data and employs optimization methods like DPO (Rafailov et al., 2024), KTO (Ethayarajh et al., 2024), IPO (Azar et al., 2024), and PRO (Song et al., 2024). Through this approach, models learn to align their outputs with verifier-provided preferences.

Various techniques are adopted to boost the foundation model's capabilities in specific areas through preference learning. For mathematical reasoning enhancement, MCTS-DPO (Xie et al., 2024) combines Monte Carlo Tree Search (Coulom, 2006; Kocsis & Szepesvári, 2006) with preference learning to generate and learn from step-level pairwise comparisons in an iterative online manner. For coding capability improvement, CodeUltraFeedback (Weyssow et al., 2024) constructs pairwise training data by using LLM verifiers to rank code outputs, then applies preference learning algorithms to optimize the model's performance. For instruction-following enhancement, Self-Rewarding (Yuan et al., 2024) enables models to generate their own verification results for creating pairwise comparison data, followed by iterative self-improvement using the DPO method.

Reinforcement Learning Reinforcement Learning optimizes models using reward signals from verifiers. Through environmental interaction and policy updates using algorithms like PPO (Schulman et al., 2017), PPO-max (Zheng et al., 2023), models iteratively improve their generation quality.

Multiple approaches are used to enhance the foundation model's capabilities in specific domains using reinforcement learning. For mathematical reasoning enhancement, Math-Shepherd (Wang et al., 2023c) implements step-wise reward mechanisms to guide progressive improvement in mathematical problem-solving capabilities. For coding capability improvement, methods like RLTF (Liu et al., 2023a) and PPOCoder (Shojaee et al., 2023a) leverage code execution results as reward signals to guide models toward more effective coding solutions. For instruction-following enhancement, approaches like InstructGPT (Ouyang et al., 2022a) and Llama (Touvron et al., 2023a;b; Dubey et al., 2024) employ reward models trained to evaluate response helpfulness, optimizing models for better instruction adherence.

5.2. Inference-based Feedback

In inference-based feedback, we modify inputs or inference strategies to obtain better outputs without changing model parameters. This approach is divided into two categories based on the visibility of verification results to the model: verifier-guided feedback and verifier-aware feedback.

Verifier-Guided In verifier-guided feedback, verifiers evaluate and select the most appropriate outputs from model-generated content without direct model interaction.

For example, Lightman et al. (2023a) and Snell et al. (2024) implement tree search algorithms guided by progress rewards, while ToT (Yao et al., 2024) employs language model verifiers to direct its tree search process. In the realm of contrastive decoding (Li et al., 2022; O'Brien & Lewis, 2023), advanced language models serve as token logits verifiers to optimize output distributions.

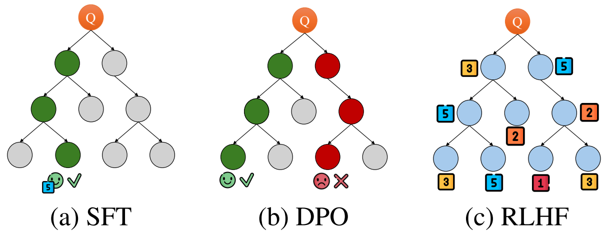

Figure 3: A verifier engineering perspective on SFT, DPO, and RLHF: gray nodes represent sample paths not used in training, while non-gray nodes represent sample paths actively used in the training process.

Verifier-Aware Verifier-Aware feedback integrates verifier feedback directly into a model's operational context to enhance the content generation process. This approach allows the model to actively consider and incorporate verifier feedback while producing its output.

Various strategies are employed to enhance specific capabilities of foundation models through verifier-aware feedback. For mathematical and coding enhancement, CRITIC (Gou et al., 2023) utilizes feedback from calculators and code interpreters to refine solutions, while Self-debug (Chen et al., 2023b) improves code quality through execution result analysis. For hallucination mitigation, approaches like ReAct (Yao et al., 2022), KGR (Guan et al., 2024), and CRITIC (Gou et al., 2023) integrate continuous feedback from search engines and knowledge graphs to ensure factual accuracy. In a similar vein, Self-Refine (Madaan et al., 2024) employs language model verifiers to iteratively improve response quality.

6. Discussion and Insights

In this section, we provide a detailed examination of the insights derived from our framework. We begin by revisiting SFT, DPO, and RLHF through the lens of verifier engineering. Subsequently, we conduct an independent analysis of each stage within the framework. Finally, we offer a systematic evaluation of potential challenges inherent to the framework as a whole.

6.1. Revisiting SFT, DPO and RLHF from Verifier Engineering

Our proposed approach to verifier engineering provides a unified perspective on commonly used post-training algorithms, offering valuable insights into their mechanisms.

SFT generates candidate responses by employing a linear search strategy that adheres to a singular search path defined by its training data. Throughout this process, the verifier classifies each token along the search path as a positive signal, while treating all other tokens as negatives. Subsequently, the foundational model is optimized through imitation learning using a cross-entropy loss function.

Similarly, DPO employs a linear search strategy, maintaining only two distinct search paths: one corresponding to the "chosen" data derived from preference pairs and the other to the "rejected" data. The verifier treats the path associated with the chosen data as positive signals and the path associated with the rejected data as negative signals. Subsequently, the foundational model is optimized through the application of pairwise loss functions.

RLHF employs a linear search strategy during the search stage, which is further enhanced by adjusting parameters such as Top-p and temperature to promote exploration within the candidate response generation process. As depicted in Figure 3(c), after search, a value function is utilized to assign scores to each state within the generation trajectory. This scoring mechanism estimates the expected return, thereby informing and guiding the optimization process through PPO algorithm.

By mapping these methods to the stages of verifier engineering, we gain a clearer understanding of how each approach in the search, verify, and feedback stage enhances foundation model capabilities.

6.2. Discussion on three stages of Verifier Engineering

The three stages of verifier engineering play a distinct and critical role in enhancing the capabilities of foundation models. This discussion delves into current challenges and proposes future research directions, focusing on search efficiency, verifier design, and the effectiveness of feedback methods.

6.2.1. Advanced Search Methods

The efficiency and effectiveness of candidate response generation are crucial for model performance. The challenge lies in balancing exploration and exploitation, as well as

	Search	**Verify**	**Feedback**	**Task**
STar (Zelikman et al., 2022a) RFT (Yuan et al., 2023c)	Linear	Golden Annotation	Imitation Learning	Math
CAG (Pan et al., 2024)	Linear	Golden Annotation	Imitation Learning	RAG
Self-Instruct (Wang et al., 2023e)	Linear	Rule-based	Imitation Learning	General
Code Alpaca (Chaudhary, 2023) WizardCoder (Luo et al., 2024d)	Linear	Rule-based	Imitation Learning	Code
ILF-Code (Chen et al., 2024a)	Linear	Rule-based & Code interpreter	Imitation Learning	Code
RAFT (Dong et al., 2023) RRHF (Yuan et al., 2023a)	Linear	ORM	Imitation Learning	General
SSO (Xiang et al., 2024)	Linear	Rule-based	Preference Learning	Alignment
CodeUltraFeedback (Weyssow et al., 2024)	Linear	Language Model	Preference Learning	Code
Self-Rewarding (Yuan et al., 2024)	Linear	Language Model	Preference Learning	Alignment
StructRAG (Li et al., 2024b)	Linear	Language Model	Preference Learning	RAG
LLAMA-BERRY (Zhang et al., 2024a)	Tree	ORM	Preference Learning	Reasoning
Math-Shepherd (Wang et al., 2024b)	Linear	Golden Annotation & Rule-based	Reinforcement Learning	Math
RLTF (Liu et al., 2023b) PPOCoder (Shojaee et al., 2023b)	Linear	Code Interpreter	Reinforcement Learning	Code
RLAIF (Lee et al., 2023)	Linear	Language Model	Reinforcement Learning	General
SIRLC (Pang et al., 2023)	Linear	Language Model	Reinforcement Learning	Reasoning
RLFH (Wen et al., 2024d)	Linear	Language Model	Reinforcement Learning	Knowledge
RLHF (Ouyang et al., 2022a)	Linear	ORM	Reinforcement Learning	Alignment
Quark (Lu et al., 2022)	Linear	Tool	Reinforcement Learning	Alignment
ReST-MCTS (Zhang et al., 2024b)	Tree	Language Model	Reinforcement Learning	Math
CRITIC (Gou et al., 2024)	Linear	Code Interpreter & Tool & Search Engine	Verifier-Aware	Math, Code & Knowledge & General
Self-Debug (Chen et al., 2023c)	Linear	Code Interpreter	Verifier-Aware	Code
Self-Refine (Madaan et al., 2023)	Linear	Language Model	Verifier-Aware	Alignment
ReAct (Yao et al., 2022)	Linear	Search Engine	Verifier-Aware	Knowledge
Constrative decoding (Li et al., 2023a)	Linear	Language Model	Verifier-Guided	General
Chain-of-verfication (Dhuliawala et al., 2023)	Linear	Language Model	Verifier-Guided	Knowledge
Inverse Value Learning (Lu et al., 2024)	Linear	Language Model	Verifier-Guided	General
PRM (Lightman et al., 2023b)	Linear	PRM	Verifier-Guided	Math
KGR (Guan et al., 2023)	Linear	Knowledge Graph	Verifier-Guided	Knowledge
UoT (Hu et al., 2024)	Tree	Language Model	Verifier-Guided	General
ToT (Yao et al., 2024)	Tree	Language Model	Verifier-Guided	Reasoning

Table 3: This paper provides a comprehensive exploration of the verifier engineering landscape, breaking it down into three core stages: search, verify, and feedback.

aligning the search process with specific optimization goals. In this subsection, we first discuss the trade-off between exploration and exploitation, and then discuss how goal-aware search can further optimize the search process.

Balancing Exploration and Exploitation The balance between exploration and exploitation is a core issue in the search stage of verifier engineering. Given an instruction, we need to search for candidate responses. For example, in DPO scenarios, we aim to obtain pair-wise responses with different verification results. Regardless of the objective, how to sample desired responses through the trade-off between exploration and exploitation remains a key challenge. Exploration refers to exploring new action spaces to obtain verified results with larger variance candidate responses, while exploitation utilizes known effective strategies to obtain responses that meet expected validation results. Exploration helps foundation models better understand the outcomes of different actions, while exploitation can obtain satisfactory responses at a lower cost. However, recent research (Wilson, 2024; Nguyen & Satoh, 2024; Murthy et al., 2024) suggests that existing methods in the post-training process often rely too heavily on exploitation while neglecting the importance of exploration. This imbalance may cause foundation models to get stuck in local optima, limiting their generalization ability. Therefore, how to better balance these two strategies during training is crucial for improving model performance.

Goal-Aware Search As shown in Section 3, existing search methods primarily rely on probability search from the next token distribution and adjust the search direction through verifier feedback. However, this approach has a fundamental issue: the current search approach often lacks a direct correlation with the optimization goal. Consequently, we cannot guide foundation models to generate candidate responses that meet our goals during the search stage and can only rely on post-hoc verification results to filter responses that align with our intentions. This delayed verification mechanism significantly reduces search efficiency and increases computational overhead. To address this challenge, search algorithms like A* (Zeng & Church, 2009) fuse the cost of the current node with heuristic estimates of the future cost to the goal. Future work can focus on developing more robust goal estimation and fusion methods.

6.2.2. Open Questions to Verifier Design

Designing effective verifiers is pivotal for enhancing the performance and reliability of foundation models. However, several unresolved issues remain regarding the optimal design and integration of verifiers. In this subsection, we first examine the considerations involved in verifier design and the need for systematic evaluation frameworks. Subsequently, we explore the complexities associated with combining multiple verifiers, highlighting the challenges that must be addressed to achieve comprehensive and scalable verification systems.

Verifier Design and Systematic Evaluation Designing specific verifiers for different types of instructions is essential. For instructions with explicit output constraints (such as grammatical format and length limitations), rule-based verifiers should be implemented due to their definitive reliability. For question-answering instructions, LLM-based verifiers are typically more effective in evaluating subjective metrics like response fluency and information content. However, recent studies (Wen et al., 2024c) have revealed only weak correlations between existing verifier evaluation metrics and downstream task performance, highlighting significant limitations in the current verifier evaluation framework. Therefore, a systematic evaluation framework is needed to comprehensively assess the effectiveness, applicable scope, and limitations of different types of verifiers. Such a framework would not only guide the selection and combination of verifiers but also establish best practices for verifier deployment across various task scenarios.

Challenges of Verifier Combinations As mentioned above, it is impossible to obtain effective verification results using a single verifier alone. Therefore, integrating multiple verifiers is essential to address the diverse requirements of candidate response evaluation. To comprehensively enhance foundation model performance across various task scenarios, an effective verifier combination system must be developed. Building an effective verifier combination system faces three key challenges:

- **Instruction Coverage**: The verifier combination must be capable of handling various types of instructions to ensure the completeness of the evaluation system. Building a comprehensive verifier framework requires a deep understanding of different task characteristics and evaluation needs, including structured output validation, open-ended question assessment, creative task evaluation, etc.
- **Automatic Routing Mechanism**: Different tasks typically require verifiers with varying forms and granularity, demanding an intelligent verifier routing system. This system needs to analyze instruction characteristics and select appropriate verifier combinations accordingly. Based on the PAC theory discussed in Section 2, the chosen verifier combinations should effectively approximate our ultimate optimization objective.
- **Verification Results Integration Strategy**: When multiple verifiers produce different verification results, a reliable decision-making mechanism is needed. This in-

cludes dynamic adjustment of different verifier weights, conflict resolution strategies, and methods for synthesizing final scores. Especially, when there are differences in verification results, factors such as the credibility and task relevance of each verifier need to be considered to make reasonable judgments.

6.2.3. Effectiveness of Feedback Methods

The effectiveness of feedback methods plays a crucial role in shaping the performance and adaptability of foundation models. In particular, we focus on two key questions: 1) whether the feedback method can accurately and efficiently improve the model's performance, and 2) whether it can generalize effectively to other queries.

Key Factors in Designing Feedback Methods Different feedback methods have different impacts on foundation model capabilities. When selecting feedback methods, it is essential to carefully balance several key factors. Firstly, the algorithm should demonstrate sufficient robustness to noise in verification results, ensuring the stability of the feedback process. Secondly, it is crucial to assess how the feedback algorithm based on specific types of verification results impacts the model. Over-optimizing certain capabilities can undermine the model's fundamental capabilities and overall generalization, potentially leading to performance degradation. Furthermore, foundation models with different capacities may require distinct optimization approaches. Larger models might benefit from more sophisticated feedback methods, while smaller models may need more conservative feedback methods to prevent capacity saturation. It is crucial to find an appropriate balance between these factors while considering the model's inherent capabilities.

Generalization over Queries Equipped with reliable verifiers and effective feedback methods, the ideal scenario is to achieve comprehensive improvements in foundation model capabilities through optimization on a limited set of queries. This necessitates feedback methods to possess strong cross-query generalization abilities. Specifically, when we enhance certain foundation model capabilities through feedback on specific queries, these improved capabilities should effectively transfer and persist when handling novel queries. However, generalization also faces significant challenges: different queries may require the model to invoke distinct capabilities, and incorrect generalization might lead to the model inappropriately applying certain capabilities in unsuitable scenarios, potentially degrading performance. Therefore, feedback methods must not only promote effective generalization but also prevent over-generalization and incorrect transfer of capabilities.

6.3. Verifier Engineering Implementation

In this section, we will discuss the problems we may meet in verifier engineering implementation in different stages to ensure efficiency, scalability, and reliability.

Search Achieving high search efficiency is a key objective in verifier engineering. Excessive search often slows down the entire verifier engineering pipeline.

To address this, most current LLM-based PPO algorithms sample only a single response for optimization. On the other hand, RLHF-like algorithms commonly incorporate importance sampling (Schulman et al., 2017; Xie et al., 2019) techniques to enhance search efficiency, minimizing the need for frequent switching between search, verify, and feedback stages while simultaneously improving sample utilization.

Verify Verifier efficiency is also a key goal in verifier engineering for giving timely and effective verification results.

When handling multiple instructions from various sources, it's crucial to employ different combinations of verifiers with different abilities to ensure accurate verification results. Determining the optimal approach to deploy all verifiers online and dynamically schedule them each time to minimize resource consumption while maximizing efficiency presents a challenging problem.

Delivering effective verification results involves addressing two major challenges: (1) ensuring the verifier's knowledge remains synchronized with the policy model, and (2) selecting the optimal verifier combination when capabilities vary or conflict across verifiers. For instance, InstructGPT (Ouyang et al., 2022a) employs a human-annotated reward model as a verifier, and to counter the limitations of a static verifier, it periodically re-annotates reward model data to align its evaluation capabilities with the evolving policy model outputs. Furthermore, Quan (2024) leverages a Mixture of Experts architecture to combine multiple verifiers with different strengths. Experiential Co-Learning (Qian et al., 2023) also draws on the knowledge of diverse foundation models to provide more robust verification results.

Feedback For highly efficient feedback, it's essential not only to enhance the feedback algorithm itself but also to optimize the entire workflow.

To increase training and inference efficiency, LoRA (Hu et al., 2021; Xin et al., 2024) improves training efficiency by reducing the number of trainable parameters and vLLM (Kwon et al., 2023) enhance inference efficiency.

To optimize the entire workflow, deciding when to apply feedback methods is crucial (Tang et al., 2024). For training-based feedback, understanding the performance gap between online and offline feedback methods is key. Research has shown that online feedback methods can maximize the capabilities of foundation models by providing timely verification results on the on-policy candidate responses, albeit requiring frequent sampling, which can be time-intensive. In contrast, offline feedback methods enable comprehensive response exploration by leveraging pre-prepared datasets for training, thereby streamlining the process. However, this approach tends to have lower data utilization efficiency. This highlights the importance of balancing online and offline feedback methods. In inference-based feedback, deciding when to call a verifier is essential. Jiang et al. (2023b) demonstrate that retrieval based on detecting uncertainty in foundation model internal states enables the model to perform better, suggesting that invoking the verifier on demand yields more effective and efficient outcomes for maximizing foundation model capabilities.

7. Conclusion

This paper introduces the concept of verifier engineering and explores the significant shift in research paradigms from feature engineering to data engineering, and ultimately to verifier engineering. Our framework provides implications and insights demonstrating that verifier engineering can optimize the capabilities of foundation models through a closed-loop feedback cycle encompassing search, verification, and feedback. Furthermore, we categorize existing search algorithms based on their granularity and schemes, review current verifiers, and classify feedback methods from both training-based and inference-based perspectives. Finally, we discuss the challenges that verifier engineering currently faces. Through this paper, we aim to stimulate further discussions and promote practical applications in the field of verifier engineering for achieving Artificial General Intelligence.

References

Anthropic. Measuring the persuasiveness of language models, 2024. https://www.anthropic.com/research/measuring-model-persuasiveness/.

Askell, A., Bai, Y., Chen, A., Drain, D., Ganguli, D., Henighan, T., Jones, A., Joseph, N., Mann, B., DasSarma, N., et al. A general language assistant as a laboratory for alignment. *ArXiv preprint*, abs/2112.00861, 2021. URL https://arxiv.org/abs/2112.00861.

Azar, M. G., Guo, Z. D., Piot, B., Munos, R., Rowland, M., Valko, M., and Calandriello, D. A general theoretical paradigm to understand learning from human preferences. In *International Conference on Artificial Intelligence and Statistics*, pp. 4447–4455. PMLR, 2024.

Bai, Y., Jones, A., Ndousse, K., Askell, A., Chen, A., DasSarma, N., Drain, D., Fort, S., Ganguli, D., Henighan, T., et al. Training a helpful and harmless assistant with reinforcement learning from human feedback. *ArXiv preprint*, abs/2204.05862, 2022a. URL https://arxiv.org/abs/2204.05862.

Bai, Y., Kadavath, S., Kundu, S., Askell, A., Kernion, J., Jones, A., Chen, A., Goldie, A., Mirhoseini, A., McKinnon, C., et al. Constitutional ai: Harmlessness from ai feedback. *ArXiv preprint*, abs/2212.08073, 2022b. URL https://arxiv.org/abs/2212.08073.

Betker, J., Goh, G., Jing, L., Brooks, T., Wang, J., Li, L., Ouyang, L., Zhuang, J., Lee, J., Guo, Y., Manassra, W., Dhariwal, P., Chu, C., Jiao, Y., and Ramesh, A. Improving image generation with better captions. URL https://api.semanticscholar.org/CorpusID:264403242.

Bradley, R. A. and Terry, M. E. Rank analysis of incomplete block designs: I. the method of paired comparisons. *Biometrika*, 39:324, 1952. URL https://api.semanticscholar.org/CorpusID:125209808.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners. In Larochelle, H., Ranzato, M., Hadsell, R., Balcan, M., and Lin, H. (eds.), *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020a. URL https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. Language models are few-shot learners, 2020b. URL https://arxiv.org/abs/2005.14165.

Burns, C., Izmailov, P., Kirchner, J. H., Baker, B., Gao, L., Aschenbrenner, L., Chen, Y., Ecoffet, A., Joglekar, M., Leike, J., Sutskever, I., and Wu, J. Weak-to-strong generalization: Eliciting strong capabilities with weak supervision, 2023. URL https://arxiv.org/abs/2312.09390.

Cai, Z., Cao, M., Chen, H., Chen, K., Chen, K., Chen, X., Chen, X., Chen, Z., Chen, Z., Chu, P., et al. Internlm2 technical report. *arXiv preprint arXiv:2403.17297*, 2024.

Cao, B., Lu, K., Lu, X., Chen, J., Ren, M., Xiang, H., Liu, P., Lu, Y., He, B., Han, X., Sun, L., Lin, H., and Yu, B. Towards scalable automated alignment of llms: A survey, 2024. URL https://arxiv.org/abs/2406.01252.

Chaudhary, S. Code alpaca: An instruction-following llama model for code generation. https://github.com/sahil280114/codealpaca, 2023.

Chen, A., Scheurer, J., Korbak, T., Campos, J. A., Chan, J. S., Bowman, S. R., Cho, K., and Perez, E. Improving code generation by training with natural language feedback, 2024a. URL https://arxiv.org/abs/2303.16749.

Chen, G., Liao, M., Li, C., and Fan, K. Alphamath almost zero: process supervision without process. *arXiv preprint arXiv:2405.03553*, 2024b.

Chen, W., Ma, X., Wang, X., and Cohen, W. W. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks, 2023a. URL https://arxiv.org/abs/2211.12588.

Chen, X., Lin, M., Schärli, N., and Zhou, D. Teaching large language models to self-debug, 2023b.

Chen, X., Lin, M., Schärli, N., and Zhou, D. Teaching large language models to self-debug, 2023c. URL https://arxiv.org/abs/2304.05128.

Chen, Z., Zhou, K., Zhao, W. X., Wan, J., Zhang, F., Zhang, D., and Wen, J.-R. Improving large language models via fine-grained reinforcement learning with minimum editing constraint, 2024c.

Christiano, P. F., Leike, J., Brown, T., Martic, M., Legg, S., and Amodei, D. Deep reinforcement learning from human preferences. *Advances in neural information processing systems*, 30, 2017.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., Hesse, C., and Schulman, J. Training verifiers to solve math word problems, 2021.

Coulom, R. Efficient selectivity and backup operators in monte-carlo tree search. In *International conference on computers and games*, pp. 72–83. Springer, 2006.

de Moura, L. M., Kong, S., Avigad, J., van Doorn, F., and von Raumer, J. The lean theorem prover (system description). In *CADE*, 2015. URL https://api.semanticscholar.org/CorpusID:232990.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pp. 248–255. Ieee, 2009.

Dhuliawala, S., Komeili, M., Xu, J., Raileanu, R., Li, X., Celikyilmaz, A., and Weston, J. Chain-of-verification reduces hallucination in large language models, 2023. URL https://arxiv.org/abs/2309.11495.

Ding, N., Chen, Y., Xu, B., Qin, Y., Zheng, Z., Hu, S., Liu, Z., Sun, M., and Zhou, B. Enhancing chat language models by scaling high-quality instructional conversations. *arXiv preprint arXiv:2305.14233*, 2023.

Dong, H., Xiong, W., Goyal, D., Zhang, Y., Chow, W., Pan, R., Diao, S., Zhang, J., SHUM, K., and Zhang, T. RAFT: Reward ranked finetuning for generative foundation model alignment. *Transactions on Machine Learning Research*, 2023. ISSN 2835-8856. URL https://openreview.net/forum?id=m7p5O7zblY.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. An image is worth 16x16 words: Transformers for image recognition at scale, 2021. URL https://arxiv.org/abs/2010.11929.

Du, Q., Zong, C., and Zhang, J. Mods: Model-oriented data selection for instruction tuning. *arXiv preprint arXiv:2311.15653*, 2023.

Dubey, A., Jauhri, A., Pandey, A., Kadian, A., Al-Dahle, A., Letman, A., Mathur, A., Schelten, A., Yang, A., Fan, A., et al. The llama 3 herd of models. *arXiv preprint arXiv:2407.21783*, 2024.

Ethayarajh, K., Xu, W., Muennighoff, N., Jurafsky, D., and Kiela, D. Kto: Model alignment as prospect theoretic optimization. *arXiv preprint arXiv:2402.01306*, 2024.

Fan, A., Lewis, M., and Dauphin, Y. Hierarchical neural story generation, 2018. URL https://arxiv.org/abs/1805.04833.

Friedland, G. and Krell, M. A capacity scaling law for artificial neural networks, 2018. URL https://arxiv.org/abs/1708.06019.

Gao, L., Madaan, A., Zhou, S., Alon, U., Liu, P., Yang, Y., Callan, J., and Neubig, G. Pal: Program-aided language models. In *International Conference on Machine Learning*, pp. 10764–10799. PMLR, 2023.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. Critic: Large language models can self-correct with tool-interactive critiquing. *arXiv preprint arXiv:2305.11738*, 2023.

Gou, Z., Shao, Z., Gong, Y., Shen, Y., Yang, Y., Duan, N., and Chen, W. Critic: Large language models can self-correct with tool-interactive critiquing, 2024. URL https://arxiv.org/abs/2305.11738.

Guan, X., Liu, Y., Lin, H., Lu, Y., He, B., Han, X., and Sun, L. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting, 2023. URL https://arxiv.org/abs/2311.13314.

Guan, X., Liu, Y., Lin, H., Lu, Y., He, B., Han, X., and Sun, L. Mitigating large language model hallucinations via autonomous knowledge graph-based retrofitting. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18126–18134, 2024.

Hearst, M. A., Dumais, S. T., Osuna, E., Platt, J., and Scholkopf, B. Support vector machines. *IEEE Intelligent Systems and their applications*, 13(4):18–28, 1998.

Hinton, G. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2015.

Holtzman, A., Buys, J., Du, L., Forbes, M., and Choi, Y. The curious case of neural text degeneration, 2020. URL https://arxiv.org/abs/1904.09751.

Hu, E. J., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. Lora: Low-rank adaptation of large language models. *arXiv preprint arXiv:2106.09685*, 2021.

Hu, Z., Liu, C., Feng, X., Zhao, Y., Ng, S.-K., Luu, A. T., He, J., Koh, P. W., and Hooi, B. Uncertainty of thoughts: Uncertainty-aware planning enhances information seeking in large language models, 2024. URL https://arxiv.org/abs/2402.03271.

Inan, H., Upasani, K., Chi, J., Rungta, R., Iyer, K., Mao, Y., Tontchev, M., Hu, Q., Fuller, B., Testuggine, D., et al. Llama guard: Llm-based input-output safeguard for human-ai conversations. *arXiv preprint arXiv:2312.06674*, 2023.

Jiang, D., Ren, X., and Lin, B. Y. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 14165–14178, Toronto, Canada, July 2023a. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.792. URL https://aclanthology.org/2023.acl-long.792.

Jiang, Z., Xu, F. F., Gao, L., Sun, Z., Liu, Q., Dwivedi-Yu, J., Yang, Y., Callan, J., and Neubig, G. Active retrieval augmented generation. *arXiv preprint arXiv:2305.06983*, 2023b.

Jordan, M. I. and Mitchell, T. M. Machine learning: Trends, perspectives, and prospects. *Science*, 349(6245):255–260, 2015.

Kocsis, L. and Szepesvári, C. Bandit based monte-carlo planning. In *European conference on machine learning*, pp. 282–293. Springer, 2006.

Krishna, K., Wieting, J., and Iyyer, M. Reformulating unsupervised style transfer as paraphrase generation. In Webber, B., Cohn, T., He, Y., and Liu, Y. (eds.), *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 737–762, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.emnlp-main.55. URL https://aclanthology.org/2020.emnlp-main.55.

Kumar, A., Zhuang, V., Agarwal, R., Su, Y., Co-Reyes, J. D., Singh, A., Baumli, K., Iqbal, S., Bishop, C., Roelofs, R., Zhang, L. M., McKinney, K., Shrivastava, D., Paduraru, C., Tucker, G., Precup, D., Behbahani, F., and Faust, A. Training language models to self-correct via reinforcement learning, 2024. URL https://arxiv.org/abs/2409.12917.

Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J. E., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

Lee, H., Phatale, S., Mansoor, H., Mesnard, T., Ferret, J., Lu, K., Bishop, C., Hall, E., Carbune, V., Rastogi, A., and Prakash, S. Rlaif: Scaling reinforcement learning from human feedback with ai feedback, 2023.

Lees, A., Tran, V. Q., Tay, Y., Sorensen, J., Gupta, J., Metzler, D., and Vasserman, L. A new generation of perspective api: Efficient multilingual character-level transformers. In *Proceedings of the 28th ACM SIGKDD conference on knowledge discovery and data mining*, pp. 3197–3207, 2022.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction. *arXiv preprint arXiv:1811.07871*, 2018a.

Leike, J., Krueger, D., Everitt, T., Martic, M., Maini, V., and Legg, S. Scalable agent alignment via reward modeling: a research direction, 2018b. URL https://arxiv.org/abs/1811.07871.

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding, 2023. URL https://arxiv.org/abs/2211.17192.

Li, X. L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., and Lewis, M. Contrastive decoding: Open-ended text generation as optimization. *arXiv preprint arXiv:2210.15097*, 2022.

Li, X. L., Holtzman, A., Fried, D., Liang, P., Eisner, J., Hashimoto, T., Zettlemoyer, L., and Lewis, M. Contrastive decoding: Open-ended text generation as optimization, 2023a. URL https://arxiv.org/abs/2210.15097.

Li, Y., Lin, Z., Zhang, S., Fu, Q., Chen, B., Lou, J.-G., and Chen, W. Making language models better reasoners with step-aware verifier. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 5315–5333, Toronto, Canada, July 2023b. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.291. URL https://aclanthology.org/2023.acl-long.291.

Li, Y., Wei, F., Zhao, J., Zhang, C., and Zhang, H. RAIN: Your language models can align themselves without finetuning. In *The Twelfth International Conference on Learning Representations*, volume abs/2309.07124, 2024a. URL https://openreview.net/forum?id=pETSfWMUzy.

Li, Z., Chen, X., Yu, H., Lin, H., Lu, Y., Tang, Q., Huang, F., Han, X., Sun, L., and Li, Y. Structrag: Boosting knowledge intensive reasoning of llms via inference-time hybrid information structurization, 2024b. URL https://arxiv.org/abs/2410.08815.

Liao, M., Luo, W., Li, C., Wu, J., and Fan, K. Mario: Math reasoning with code interpreter output – a reproducible pipeline, 2024.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step. *arXiv preprint arXiv:2305.20050*, 2023a.

Lightman, H., Kosaraju, V., Burda, Y., Edwards, H., Baker, B., Lee, T., Leike, J., Schulman, J., Sutskever, I., and Cobbe, K. Let's verify step by step, 2023b.

Lin, H., Sun, Z., Yang, Y., and Welleck, S. Lean-star: Learning to interleave thinking and proving, 2024.

Liu, J., Zhu, Y., Xiao, K., Fu, Q., Han, X., Yang, W., and Ye, D. Rltf: Reinforcement learning from unit test feedback. *arXiv preprint arXiv:2307.04349*, 2023a.

Liu, J., Zhu, Y., Xiao, K., Fu, Q., Han, X., Yang, W., and Ye, D. Rltf: Reinforcement learning from unit test feedback, 2023b. URL https://arxiv.org/abs/2307.04349.

Liu, M., Zhu, M., and Zhang, W. Goal-conditioned reinforcement learning: Problems and solutions, 2022. URL https://arxiv.org/abs/2201.08299.

Liu, W., Zeng, W., He, K., Jiang, Y., and He, J. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*, 2024. URL https://openreview.net/forum?id=BTKAeLqLMw.

Lu, X., Welleck, S., Hessel, J., Jiang, L., Qin, L., West, P., Ammanabrolu, P., and Choi, Y. Quark: Controllable text generation with reinforced unlearning, 2022.

Lu, X., Wen, X., Lu, Y., Yu, B., Lin, H., Yu, H., Sun, L., Han, X., and Li, Y. Transferable post-training via inverse value learning, 2024. URL https://arxiv.org/abs/2410.21027.

Luo, H., Sun, Q., Xu, C., Zhao, P., Lou, J., Tao, C., Geng, X., Lin, Q., Chen, S., and Zhang, D. Wizardmath: Empowering mathematical reasoning for large language models via reinforced evol-instruct, 2023a.

Luo, L., Liu, Y., Liu, R., Phatale, S., Lara, H., Li, Y., Shu, L., Zhu, Y., Meng, L., Sun, J., and Rastogi, A. Improve mathematical reasoning in language models by automated process supervision, 2024a. URL https://arxiv.org/abs/2406.06592.

Luo, L., Liu, Y., Liu, R., Phatale, S., Lara, H., Li, Y., Shu, L., Zhu, Y., Meng, L., Sun, J., et al. Improve mathematical reasoning in language models by automated process supervision. *arXiv preprint arXiv:2406.06592*, 2024b.

Luo, X., Zhu, Q., Zhang, Z., Qin, L., Zhang, X., Yang, Q., Xu, D., and Che, W. Python is not always the best choice: Embracing multilingual program of thoughts, 2024c. URL https://arxiv.org/abs/2402.10691.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. Wizardcoder: Empowering code large language models with evol-instruct. *arXiv preprint arXiv:2306.08568*, 2023b.

Luo, Z., Xu, C., Zhao, P., Sun, Q., Geng, X., Hu, W., Tao, C., Ma, J., Lin, Q., and Jiang, D. Wizardcoder: Empowering code large language models with evol-instruct. In *The Twelfth International Conference on Learning Representations*, 2024d. URL https://openreview.net/forum?id=UnUwSIgK5W.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., Gupta, S., Majumder, B. P., Hermann, K., Welleck, S., Yazdanbakhsh, A., and Clark, P. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, volume 36, pp. 46534–46594, 2023. URL https://openreview.net/forum?id=S37hOerQLB.

Madaan, A., Tandon, N., Gupta, P., Hallinan, S., Gao, L., Wiegreffe, S., Alon, U., Dziri, N., Prabhumoye, S., Yang, Y., et al. Self-refine: Iterative refinement with self-feedback. *Advances in Neural Information Processing Systems*, 36, 2024.

Markov, T., Zhang, C., Agarwal, S., Nekoul, F. E., Lee, T., Adler, S., Jiang, A., and Weng, L. A holistic approach to undesired content detection in the real world. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pp. 15009–15018, 2023.

Mo, S. and Xin, M. Tree of uncertain thoughts reasoning for large language models, 2023. URL https://arxiv.org/abs/2309.07694.

Mu, T., Helyar, A., Heidecke, J., Achiam, J., Vallone, A., Kivlichan, I., Lin, M., Beutel, A., Schulman, J., and Weng, L. Rule based rewards for language model safety. *Advances in Neural Information Processing Systems*, 2024.

Murthy, R., Heinecke, S., Niebles, J. C., Liu, Z., Xue, L., Yao, W., Feng, Y., Chen, Z., Gokul, A., Arpit, D., Xu, R., Mui, P., Wang, H., Xiong, C., and Savarese, S. Rex: Rapid exploration and exploitation for ai agents, 2024. URL https://arxiv.org/abs/2307.08962.

Nakano, R., Hilton, J., Balaji, S., Wu, J., Ouyang, L., Kim, C., Hesse, C., Jain, S., Kosaraju, V., Saunders, W., et al. Webgpt: Browser-assisted question-answering with human feedback. *arXiv preprint arXiv:2112.09332*, 2021.

Nguyen, H.-T. and Satoh, K. Balancing exploration and exploitation in llm using soft rllf for enhanced negation understanding. *arXiv preprint arXiv:2403.01185*, 2024.

O'Brien, S. and Lewis, M. Contrastive decoding improves reasoning in large language models. *arXiv preprint arXiv:2309.09117*, 2023.

Olausson, T., Gu, A., Lipkin, B., Zhang, C., Solar-Lezama, A., Tenenbaum, J., and Levy, R. Linc: A neurosymbolic approach for logical reasoning by combining language models with first-order logic provers. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pp. 5153–5176. Association for Computational Linguistics, 2023. doi: 10.18653/v1/2023.emnlp-main.313. URL http://dx.doi.org/10.18653/v1/2023.emnlp-main.313.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., et al. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744, 2022a.

Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. Training language models to follow instructions with human feedback, 2022b. URL https://arxiv.org/abs/2203.02155.

Pan, R., Cao, B., Lin, H., Han, X., Zheng, J., Wang, S., Cai, X., and Sun, L. Not all contexts are equal: Teaching llms credibility-aware generation, 2024. URL https://arxiv.org/abs/2404.06809.

Pang, J.-C., Wang, P., Li, K., Chen, X.-H., Xu, J., Zhang, Z., and Yu, Y. Language model self-improvement by reinforcement learning contemplation, 2023. URL https://arxiv.org/abs/2305.14483.

Peng, B., Li, C., He, P., Galley, M., and Gao, J. Instruction tuning with gpt-4, 2023.

Plappert, M., Andrychowicz, M., Ray, A., McGrew, B., Baker, B., Powell, G., Schneider, J., Tobin, J., Chociej, M., Welinder, P., Kumar, V., and Zaremba, W. Multi-goal reinforcement learning: Challenging robotics environments and request for research, 2018. URL https://arxiv.org/abs/1802.09464.

Qian, C., Dang, Y., Li, J., Liu, W., Xie, Z., Wang, Y., Chen, W., Yang, C., Cong, X., Che, X., et al. Experiential co-learning of software-developing agents. *arXiv preprint arXiv:2312.17025*, 2023.

Quan, S. Dmoerm: Recipes of mixture-of-experts for effective reward modeling. *arXiv preprint arXiv:2403.01197*, 2024.

Quinlan, J. R. Induction of decision trees. *Machine learning*, 1:81–106, 1986.

Rafailov, R., Sharma, A., Mitchell, E., Manning, C. D., Ermon, S., and Finn, C. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 36, 2024.

Robertson, S., Zaragoza, H., et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

Saunders, W., Yeh, C., Wu, J., Bills, S., Ouyang, L., Ward, J., and Leike, J. Self-critiquing models for assisting human evaluators. *ArXiv preprint*, abs/2206.05802, 2022. URL https://arxiv.org/abs/2206.05802.

Schaul, T., Horgan, D., Gregor, K., and Silver, D. Universal value function approximators. In Bach, F. and Blei, D. (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 1312–1320, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/schaul15.html.

Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. ISSN 0893-6080. doi: https://doi.org/10.1016/j.neunet.2014.09.003. URL https://www.sciencedirect.com/science/article/pii/S0893608014002135.

Schulman, J., Wolski, F., Dhariwal, P., Radford, A., and Klimov, O. Proximal policy optimization algorithms, 2017.

Sevilla, J., Heim, L., Ho, A., Besiroglu, T., Hobbhahn, M., and Villalobos, P. Compute trends across three eras of machine learning. In *2022 International Joint Conference on Neural Networks (IJCNN)*, pp. 1–8. IEEE, July 2022. doi: 10.1109/ijcnn55064.2022.9891914. URL http://dx.doi.org/10.1109/IJCNN55064.2022.9891914.

Shojaee, P., Jain, A., Tipirneni, S., and Reddy, C. K. Execution-based code generation using deep reinforcement learning. *arXiv preprint arXiv:2301.13816*, 2023a.

Shojaee, P., Jain, A., Tipirneni, S., and Reddy, C. K. Execution-based code generation using deep reinforcement learning, 2023b. URL https://arxiv.org/abs/2301.13816.

Snell, C., Lee, J., Xu, K., and Kumar, A. Scaling llm test-time compute optimally can be more effective than scaling model parameters. *arXiv preprint arXiv:2408.03314*, 2024.

Song, F., Yu, B., Li, M., Yu, H., Huang, F., Li, Y., and Wang, H. Preference ranking optimization for human alignment. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pp. 18990–18998, 2024.

Stiennon, N., Ouyang, L., Wu, J., Ziegler, D., Lowe, R., Voss, C., Radford, A., Amodei, D., and Christiano, P. F. Learning to summarize with human feedback. *Advances in Neural Information Processing Systems*, 33: 3008–3021, 2020.

Sun, H., Haider, M., Zhang, R., Yang, H., Qiu, J., Yin, M., Wang, M., Bartlett, P., and Zanette, A. Fast best-of-n decoding via speculative rejection, 2024a. URL https://arxiv.org/abs/2410.20290.

Sun, S., Li, J., Yuan, W., Yuan, R., Li, W., and Liu, P. The critique of critique. *ArXiv preprint*, abs/2401.04518, 2024b. URL https://arxiv.org/abs/2401.04518.

Tang, Y., Guo, D. Z., Zheng, Z., Calandriello, D., Cao, Y., Tarassov, E., Munos, R., Pires, B. Á., Valko, M., Cheng, Y., et al. Understanding the performance gap between online and offline alignment algorithms. *arXiv preprint arXiv:2405.08448*, 2024.

Touvron, H., Lavril, T., Izacard, G., Martinet, X., Lachaux, M.-A., Lacroix, T., Rozière, B., Goyal, N., Hambro, E., Azhar, F., et al. Llama: Open and efficient foundation language models. *arXiv preprint arXiv:2302.13971*, 2023a.

Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., et al. Llama 2: Open foundation and fine-tuned chat models. *arXiv preprint arXiv:2307.09288*, 2023b.

Tsai, C. F., Zhou, X., Liu, S. S., Li, J., Yu, M., and Mei, H. Can large language models play text games well? current state-of-the-art and open questions. *arXiv preprint arXiv:2304.02868*, 2023.

Vapnik, V. N. *The Nature of Statistical Learning Theory*. Springer, 2000.

Vijayakumar, A. K., Cogswell, M., Selvaraju, R. R., Sun, Q., Lee, S., Crandall, D., and Batra, D. Diverse beam search: Decoding diverse solutions from neural sequence models, 2018. URL https://arxiv.org/abs/1610.02424.

Wang, K., Ren, H., Zhou, A., Lu, Z., Luo, S., Shi, W., Zhang, R., Song, L., Zhan, M., and Li, H. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning. *arXiv preprint arXiv:2310.03731*, 2023a.

Wang, K., Ren, H., Zhou, A., Lu, Z., Luo, S., Shi, W., Zhang, R., Song, L., Zhan, M., and Li, H. Mathcoder: Seamless code integration in llms for enhanced mathematical reasoning, 2023b. URL https://arxiv.org/abs/2310.03731.

Wang, P., Li, L., Shao, Z., Xu, R., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: A label-free step-by-step verifier for llms in mathematical reasoning. *arXiv preprint arXiv:2312.08935*, 2023c.

Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024a.

Wang, P., Li, L., Shao, Z., Xu, R. X., Dai, D., Li, Y., Chen, D., Wu, Y., and Sui, Z. Math-shepherd: Verify and reinforce llms step-by-step without human annotations, 2024b. URL https://arxiv.org/abs/2312.08935.

Wang, X. and Zhou, D. Chain-of-thought reasoning without prompting. *ArXiv preprint*, abs/2402.10200, 2024. URL https://arxiv.org/abs/2402.10200.

Wang, X., Wei, J., Schuurmans, D., Le, Q. V., Chi, E. H., Narang, S., Chowdhery, A., and Zhou, D. Self-consistency improves chain of thought reasoning in language models. In *The Eleventh International Conference on Learning Representations*, 2023d. URL https://openreview.net/forum?id=1PL1NIMMrw.

Wang, Y., Kordi, Y., Mishra, S., Liu, A., Smith, N. A., Khashabi, D., and Hajishirzi, H. Self-instruct: Aligning language models with self-generated instructions. In Rogers, A., Boyd-Graber, J., and Okazaki, N. (eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pp. 13484–13508, Toronto, Canada, 2023e. Association for Computational Linguistics. doi: 10.18653/v1/2023.acl-long.754. URL https://aclanthology.org/2023.acl-long.754.

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Xia, F., Chi, E., Le, Q. V., Zhou, D., et al. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35:24824–24837, 2022.

Wen, J., Zhong, R., Khan, A., Perez, E., Steinhardt, J., Huang, M., Boman, S. R., He, H., and Feng, S. Language models learn to mislead humans via rlhf. *arXiv preprint arXiv:2409.12822*, 2024a.

Wen, J., Zhong, R., Khan, A., Perez, E., Steinhardt, J., Huang, M., Bowman, S. R., He, H., and Feng, S. Language models learn to mislead humans via rlhf, 2024b. URL https://arxiv.org/abs/2409.12822.

Wen, X., Lou, J., Lu, Y., Lin, H., Yu, X., Lu, X., He, B., Han, X., Zhang, D., and Sun, L. Rethinking reward model evaluation: Are we barking up the wrong tree? *arXiv preprint arXiv:2410.05584*, 2024c.

Wen, X., Lu, X., Guan, X., Lu, Y., Lin, H., He, B., Han, X., and Sun, L. On-policy fine-grained knowledge feedback for hallucination mitigation, 2024d. URL https://arxiv.org/abs/2406.12221.

Weyssow, M., Kamanda, A., and Sahraoui, H. Codeultrafeedback: An llm-as-a-judge dataset for aligning large language models to coding preferences, 2024.

Wilson, D. Llm tree search. *arXiv preprint arXiv:2410.19117*, 2024.

Wu, T., Yuan, W., Golovneva, O., Xu, J., Tian, Y., Jiao, J., Weston, J., and Sukhbaatar, S. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*, 2024.

Xiang, H., Yu, B., Lin, H., Lu, K., Lu, Y., Han, X., Sun, L., Zhou, J., and Lin, J. Aligning large language models via self-steering optimization, 2024. URL https://arxiv.org/abs/2410.17131.

Xie, T., Ma, Y., and Wang, Y.-X. Towards optimal off-policy evaluation for reinforcement learning with marginalized importance sampling. *Advances in neural information processing systems*, 32, 2019.

Xie, Y., Goyal, A., Zheng, W., Kan, M.-Y., Lillicrap, T. P., Kawaguchi, K., and Shieh, M. Monte carlo tree search boosts reasoning via iterative preference learning. *arXiv preprint arXiv:2405.00451*, 2024.

Xin, C., Lu, Y., Lin, H., Zhou, S., Zhu, H., Wang, W., Liu, Z., Han, X., and Sun, L. Beyond full fine-tuning: Harnessing the power of LoRA for multi-task instruction tuning. In Calzolari, N., Kan, M.-Y., Hoste, V., Lenci, A., Sakti, S., and Xue, N. (eds.), *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pp. 2307–2317, Torino, Italia, May 2024. ELRA and ICCL. URL https://aclanthology.org/2024.lrec-main.206.

Xu, C., Guo, D., Duan, N., and McAuley, J. Baize: An open-source chat model with parameter-efficient tuning on self-chat data. *arXiv preprint arXiv:2304.01196*, 2023.

Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., Narasimhan, K. R., and Cao, Y. React: Synergizing reasoning and acting in language models. In *The Eleventh International Conference on Learning Representations*, 2022.

Yao, S., Yu, D., Zhao, J., Shafran, I., Griffiths, T., Cao, Y., and Narasimhan, K. Tree of thoughts: Deliberate problem solving with large language models. *Advances in Neural Information Processing Systems*, 36, 2024.

Yu, F., Gao, A., and Wang, B. OVM, outcome-supervised value models for planning in mathematical reasoning. In Duh, K., Gomez, H., and Bethard, S. (eds.), *Findings of the Association for Computational Linguistics: NAACL 2024*, pp. 858–875, Mexico City, Mexico, June 2024. Association for Computational Linguistics. doi: 10.18653/v1/2024.findings-naacl.55. URL https://aclanthology.org/2024.findings-naacl.55.

Yu, L., Jiang, W., Shi, H., Yu, J., Liu, Z., Zhang, Y., Kwok, J. T., Li, Z., Weller, A., and Liu, W. Metamath: Bootstrap your own mathematical questions for large language models. *arXiv preprint arXiv:2309.12284*, 2023a.

Yu, Z., Zhang, X., Shang, N., Huang, Y., Xu, C., Zhao, Y., Hu, W., and Yin, Q. Wavecoder: Widespread and versatile enhanced instruction tuning with refined data generation. *arXiv preprint arXiv:2312.14187*, 2023b.

Yuan, H., Yuan, Z., Tan, C., Wang, W., Huang, S., and Huang, F. RRHF: Rank responses to align language models with human feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*, 2023a. URL https://openreview.net/forum?id=EdIGMCHk4l.

Yuan, W., Pang, R. Y., Cho, K., Sukhbaatar, S., Xu, J., and Weston, J. Self-rewarding language models. *ArXiv preprint*, abs/2401.10020, 2024. URL https://arxiv.org/abs/2401.10020.

Yuan, Z., Yuan, H., Li, C., Dong, G., Lu, K., Tan, C., Zhou, C., and Zhou, J. Scaling relationship on learning mathematical reasoning with large language models, 2023b.

Yuan, Z., Yuan, H., Li, C., Dong, G., Lu, K., Tan, C., Zhou, C., and Zhou, J. Scaling relationship on learning mathematical reasoning with large language models, 2023c. URL https://arxiv.org/abs/2308.01825.

Yue, X., Qu, X., Zhang, G., Fu, Y., Huang, W., Sun, H., Su, Y., and Chen, W. Mammoth: Building math generalist models through hybrid instruction tuning. *arXiv preprint arXiv:2309.05653*, 2023.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. In Koyejo, S., Mohamed, S., Agarwal, A., Belgrave, D., Cho, K., and Oh, A. (eds.), *Advances in Neural Information Processing Systems*, volume 35, pp. 15476–15488. Curran Associates, Inc., 2022a. URL https://proceedings.neurips.cc/paper_files/paper/2022/file/639a9a172c044fbb64175b5fad42e9a5-Paper-Conference.pdf.

Zelikman, E., Wu, Y., Mu, J., and Goodman, N. Star: Bootstrapping reasoning with reasoning. *Advances in Neural Information Processing Systems*, 35:15476–15488, 2022b.

Zelikman, E., Harik, G., Shao, Y., Jayasiri, V., Haber, N., and Goodman, N. D. Quiet-star: Language models can teach themselves to think before speaking. *ArXiv preprint*, abs/2403.09629, 2024. URL https://arxiv.org/abs/2403.09629.

Zeng, W. and Church, R. L. Finding shortest paths on real road networks: the case for a*. *Int. J. Geogr. Inf. Sci.*, 23(4):531–543, April 2009. ISSN 1365-8816. doi: 10.1080/13658810801949850. URL https://doi.org/10.1080/13658810801949850.

Zhang, D., Wu, J., Lei, J., Che, T., Li, J., Xie, T., Huang, X., Zhang, S., Pavone, M., Li, Y., Ouyang, W., and Zhou, D. Llama-berry: Pairwise optimization for o1-like olympiad-level mathematical reasoning, 2024a. URL https://arxiv.org/abs/2410.02884.

Zhang, D., Zhoubian, S., Hu, Z., Yue, Y., Dong, Y., and Tang, J. Rest-mcts*: Llm self-training via process reward guided tree search, 2024b. URL https://arxiv.org/abs/2406.03816.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction. *arXiv preprint arXiv:2408.15240*, 2024c.

Zhang, L., Hosseini, A., Bansal, H., Kazemi, M., Kumar, A., and Agarwal, R. Generative verifiers: Reward modeling as next-token prediction, 2024d. URL https://arxiv.org/abs/2408.15240.

Zheng, L., Chiang, W.-L., Sheng, Y., Zhuang, S., Wu, Z., Zhuang, Y., Lin, Z., Li, Z., Li, D., Xing, E., et al. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623, 2024a. URL https://proceedings.neurips.cc/paper_files/paper/2023/hash/91f18a1287b398d378ef22505bf41832-Abstract-Datasets_and_Benchmarks.html.

Zheng, R., Dou, S., Gao, S., Hua, Y., Shen, W., Wang, B., Liu, Y., Jin, S., Liu, Q., Zhou, Y., Xiong, L., Chen, L., Xi, Z., Xu, N., Lai, W., Zhu, M., Chang, C., Yin, Z., Weng, R., Cheng, W., Huang, H., Sun, T., Yan, H., Gui, T., Zhang, Q., Qiu, X., and Huang, X. Secrets of rlhf in large language models part i: Ppo, 2023.

Zheng, T., Zhang, G., Shen, T., Liu, X., Lin, B. Y., Fu, J., Chen, W., and Yue, X. Opencodeinterpreter: Integrating code generation with execution and refinement, 2024b.