

Nanbeige4-3B Technical Report: Exploring the Frontier of Small Language Models

Nanbeige LLM Lab, Boss Zhipin

Abstract

We present Nanbeige4-3B, a family of small-scale but high-performing language models. Pretrained on 23T high-quality tokens and finetuned on over 30 million diverse instructions, we extend the boundary of the scaling law for small language models. In pre-training, we design a Fine-Grained Warmup-Stable-Decay (FG-WSD) training scheduler, which progressively refines data mixtures across stages to boost model performance. In post-training, to improve the quality of the SFT data, we design a joint mechanism that integrates deliberative generation refinement and chain-of-thought reconstruction, yielding substantial gains on complex tasks. Following SFT, we employ our flagship reasoning model to distill Nanbeige4-3B through our proposed Dual Preference Distillation (DPD) method, which leads to further performance gains. Finally, a multi-stage reinforcement learning phase was applied, leveraging verifiable rewards and preference modeling to strengthen abilities on both reasoning and human alignment. Extensive evaluations show that Nanbeige4-3B not only significantly outperforms models of comparable parameter scale but also rivals much larger models across a wide range of benchmarks. The model checkpoints are available at <https://huggingface.co/Nanbeige>.

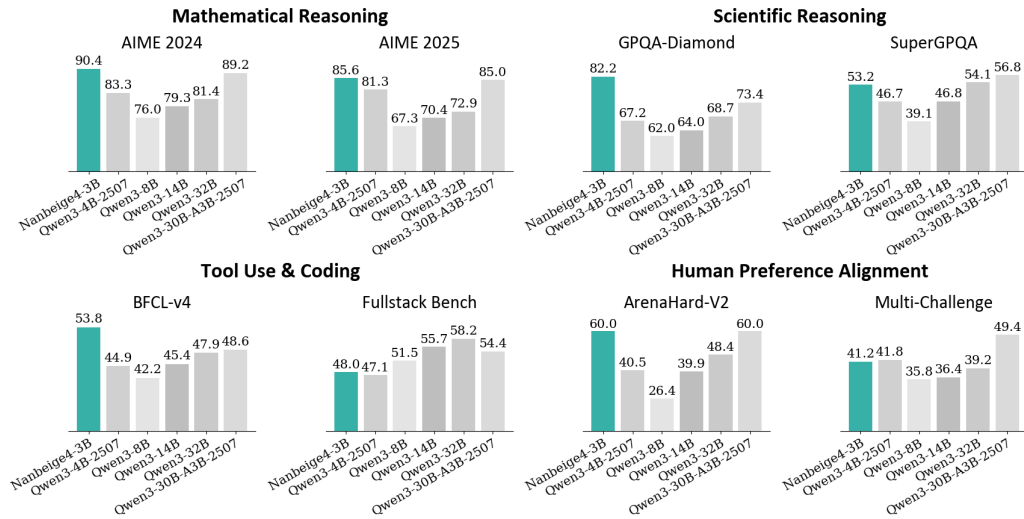


Figure 1: Performance Comparison between Nanbeige4-3B-Thinking and Qwen series models.

1 Introduction

In recent years, we have witnessed the emergence of capable large-scale language models ranging from hundreds of billions to trillions of parameters [43, 46, 11, 32]. They have demonstrated remarkable reasoning abilities and have become a pivotal driving force in the evolution of artificial intelligence. Despite their impressive performance, these models come with substantial inference costs in deployment and high training expenses for research, whether for full-scale replication or fine-tuning. This scenario underscores the importance of exploring the potential of Small Language Models (SLMs) as a resource-efficient alternative.

In this work, we present the Nanbeige4-3B model family. Despite their compact size, they exhibit remarkably strong and well-balanced capabilities in mathematics, scientific reasoning, human preference alignment, creative writing, and tool use—substantially outperforming many larger models. These results highlight the effectiveness of our training methodology and clearly demonstrate that **small, well-engineered models can achieve performance that surpasses far larger models**.

To provide a clear sense of the model’s capability, we compare Nanbeige4-3B against the Qwen3 series across multiple parameter scales. For the base models, we conduct several post-training runs using identical SFT datasets, where Nanbeige4-3B-Base significantly outperforms Qwen3-8B-Base. For the reasoning model, Nanbeige4-3B-Thinking demonstrates overall performance that surpasses Qwen3-14B on average. Notably, it even outperforms the substantially larger Qwen3-32B and Qwen3-30B-A3B on mathematical and scientific reasoning, as well as on specific dimensions concerning tool use and human preference alignment. Moreover, in terms of creative writing ability, Nanbeige4-3B-Thinking approaches the performance of several state-of-the-art large models, as elaborated in the WritingBench Leaderboard November 2025.

Table 1: Performance comparison between Nanbeige4-3B-Thinking and other large parameter-scale models on WritingBench [41]. Scores are taken from the official leaderboard.

Model	Overall	Academic & Engineering	Finance & Business	Politics & Law	Literature & Arts	Education	Advertising & Marketing
GPT-5	83.87	84.46	83.78	82.21	85.26	85.64	82.34
Qwen-235B-A22B-Thinking	82.34	83.43	83.44	79.67	82.68	84.52	80.94
Doubao-Seed-1.6-Thinking	79.29	80.79	79.74	78.05	78.37	81.48	77.95
Gemini-2.5-Pro	79.26	79.59	78.24	78.69	80.92	81.30	77.27
Nanbeige4-3B-Thinking-2511	79.03	80.77	82.37	77.41	76.05	80.81	76.55
Deepseek-R1-0528	78.92	78.79	78.38	77.27	80.91	80.41	78.44
Grok-4	74.65	75.44	73.99	73.08	74.66	77.48	74.73
O4-mini	72.90	76.04	73.19	71.26	69.87	75.91	72.66
O3-mini	68.02	69.52	68.61	66.92	66.54	71.48	65.95

In this report, we present a comprehensive overview of the model training pipeline—from the pre-training to the post-training process, and highlight the techniques that most substantially contribute to our model’s performance improvements. Specifically, Nanbeige4-3B is featured with the following technologies to improve model capabilities:

Pre-training with Hybrid Data Filtering and Fine-Grained WSD.

- **Hybrid Data Filtering.** To enable more precise filtering of high-quality data, we develop a hybrid strategy combining tagging-based scoring [29, 47] with retrieval-based recalling [7], obtaining the final comprehensive training corpus consists of 23 trillion tokens high-quality data.
- **Fine-Grained WSD.** We introduce a Fine-Grained Warmup-Stable-Decay (FG-WSD) scheduler to maximize the utility of high-quality data. This approach employs a fine-grained and quality-progressive data curriculum, in which the stable stage is partitioned into multiple phases with progressively refined data mixtures. Evaluations demonstrate that our method yields substantial improvements over the vanilla WSD scheduler [13, 19].

Post-training with Multi-Stage SFT, Distillation, and Reinforcement Learning.

- **Multi-Stage SFT.** We first fine-tune our base model on over 30 million cleaned samples related to math, science, and code to establish strong reasoning capabilities. Subsequently, we apply

curriculum learning with increasingly diverse and more challenging instructions to ensure robust performance across a wide range of tasks. To further improve the quality of SFT responses, we introduce an innovative approach that combines deliberative learning with Chain-of-Thought (CoT) reconstruction: we first refine the Solution component to make it superior, then reconstruct a corresponding CoT that logically leads to this improved solution. This yields high-quality SFT training examples that significantly outperform those generated via rejection sampling.

- **Distillation.** Following SFT, we employ the Nanbeige flagship reasoning model as the teacher to distill the Nanbeige4-3B student model, with our proposed Dual Preference Distillation (DPD) method. In particular, we innovate the loss function design: on the one hand, the student model learns to mimic the teacher’s output distribution as a policy model; on the other hand, it is simultaneously trained to distinguish between high-quality and low-quality responses.
- **Reinforcement Learning.** Building upon the distilled model, we conduct multi-stage RL training to further boost performance. We design a suite of reward and verification strategies tailored to different training phases—including STEM reasoning, coding, and human preference alignment—ensuring consistent and stable improvements throughout.

Building on the techniques described above, we develop a base model Nanbeige4-3B-Base, and a reasoning-enhanced model Nanbeige4-3B-Thinking. To support the community and foster open, reproducible research in LLMs, we **open-source the Nanbeige4-3B model family**. We hope this will empower researchers and developers to explore advanced training methodologies, accelerate innovation in reasoning-centric models, and contribute to a more open and collaborative AI ecosystem.

2 Pre-Training

In this section, we first describe the construction of our pre-training dataset and the corresponding training recipe, followed by an evaluation of the Nanbeige4-3B-Base model’s performance.

2.1 Pre-Training Data

During the construction of the pre-training corpus for Nanbeige4-3B, we extensively collect high-quality and diverse data. Our corpus encompasses a diverse range of web pages, scholarly articles, books, source code, and other materials. To strengthen the grasp of the human world, we not only extract clean text from HTML documents but also develop a highly efficient PDF text-extraction pipeline. We further augment the pre-training mix with synthetic data that target specialized competencies: these take the form of question-answer pairs, textbooks, lecture notes, and long chain-of-thought samples, which constitute 15% of the total pre-training tokens.

2.2 Pre-Training Recipe

In developing the training recipe for Nanbeige4-3B, we focus on two key issues:

1. How to identify high-quality data while filtering out low-quality samples.
2. How to make full use of the selected data to further improve performance.

In the subsequent parts of this section, we first introduce how data quality is assessed and annotated, and then present how these quality ratings are utilized during training to enhance model performance.

2.2.1 Data Quality Identify

To retain high-quality samples while filtering out low-quality ones, we assess the quality of each data entry in the corpus using two complementary strategies: (1) multi-dimensional tagging that inspects intrinsic attributes, and (2) similarity-based scoring against a curated set of high-quality seed data to assess extrinsic alignment.

Multi-dimensional tagging. We adopt a similar workflow as in prior research on pre-training data quality assessment [13, 29]. This involves constructing a comprehensive labeling system, sampling and annotating data using a strong model, distilling the annotations to a smaller model for scalable annotation, and finally applying weighted ranking to select high-quality samples. Our labeling framework spans two key aspects: text format and content. Initially, we define over 60 dimensions,

including knowledge density, reasoning density, and text fluency, etc. These are later consolidated into a final set of 20 dimensions based on similarity. Empirical results validate two key insights: content-related labels are more predictive of quality than format-related ones, and a fine-grained zero-to-nine scoring scale yields more accurate data selection than a binary zero-or-one scale.

Similarity-based scoring. We build a retrieval database containing hundreds of billions of entries, supporting efficient hybrid text-based and vector-based retrieval. With this robust retrieval infrastructure, we continuously iterate on seed data curation, as well as the retrieval methodologies. For the seed data, we prioritize samples that rank high within our quality tagging framework while ensuring data provenance from reliable and authoritative sources. This dual-criteria approach guarantees that our foundational dataset maintains both exceptional quality standards and trustworthy origins. For the retrieval methodology, we conduct extensive experiments on how to balance between similarity scores and quality assessments. We discover that applying retrieval strategies on top of our quality labeling system enables more precise identification and selection of high-quality data.

By combining multi-dimensional tagging with similarity-based scoring, we filter out tens of trillions of tokens that do not meet our criteria and retain 12.5 trillion high-quality training data. From these 12.5 tokens, we further select 6.5 trillion tokens of even higher quality for up-sampling two or more epochs, ultimately forming our final 23 trillion token training corpus.

2.2.2 Data Utility Scheduler

To fully leverage the selected data for further performance improvement, we introduce a novel learning rate scheduler named Fine-Grained Warmup-Stable-Decay (FG-WSD). This scheduler increases the learning rate during an initial warm-up phase, maintains it across multiple carefully designed stable stages, and finally applies a smooth decay. The multi-stage stable phases provide greater room for exploration based on the previously obtained data scores, while preventing the undesirable coupling between data ordering and learning rate changes.

Cosine Decay vs. WSD vs. FG-WSD Pre-training learning rate schedulers generally fall into two categories: warmup-cosine-decay [34, 35, 9] and warmup-stable-decay (WSD) [13, 19, 20]. Through experimental validation, we find that when the data quality during the annealing phase is sufficiently high, warmup-stable-decay significantly outperforms warmup-cosine-decay. Consequently, we adopt the WSD approach as our foundational scheduling strategy. Building upon this baseline, we further introduce an enhanced variant: FG-WSD (Fine-Grained WSD). Recognizing that the WSD schedule maintains a constant learning rate during the stable phase, we optimize the data resampling strategy accordingly. Rather than uniformly sampling high-quality data throughout the entire training process, FG-WSD divides training into multiple fine-grained stages and progressively increases the proportion of higher-quality data mixtures in later stages.

Preliminary Experiments. We verify the effectiveness of FG-WSD on a 1B-parameter model with a fixed 100B-token corpus in the decay stage. The stable phase consumes one epoch of 500B medium-quality (MQ) tokens and two epochs of 250B high-quality (HQ) tokens, for a total of 1T stable-phase tokens. Under vanilla WSD, these 1T tokens are shuffled uniformly: every sample is drawn from a static 1:1 mixture of HQ and MQ data. FG-WSD, by contrast, splits the stable phase into two contiguous stages. Stage 1 processes 750B tokens composed of one epoch of HQ data (250B tokens) plus one epoch of MQ data (500B tokens). Stage 2 then continues for an additional 250B-token containing only HQ subset.

As shown in Table 2, our FG-WSD method outperforms the vanilla WSD approach across all benchmarks¹. Notably, the improvement is more pronounced on mathematical and reasoning tasks (e.g., GSM8K [4], CMATH [40], and BBH [30]) compared to knowledge and science benchmarks (e.g., MMLU [12], CMMLU [15], and MMLU-Pro [38]). This is because our quality scoring framework prioritizes reasoning density over knowledge density during high-quality data selection.

After validating the effectiveness of FG-WSD with a 1B-parameter model trained on 1T tokens, we scale up the method to the full training corpus of Nanbeige4-3B-Base. Specifically, the training process is divided into four distinct stages: **Warmup, Diversity-Enriched Stable, High-Quality**

¹For GSM8k, Cmath, and BBH, we run 3-shot evaluation. For MMLU, CMMLU, and MMLU-Pro, we run 5-shot evaluation.

Table 2: Performance comparison between vanilla WSD scheduler and our proposed Fine-Grained WSD scheduler. The experiment is implemented on 1B parameter scale model with 1T tokens.

Learning Rate Scheduler	GSM8k	Cmath	BBH	MMLU	CMMLU	MMLU-Pro
Vanilla WSD	27.1	34.5	29.3	49.2	50.3	16.87
Fine-Grained WSD	34.3	39.5	31.6	50.6	51.9	18.64

Table 3: Training stages of the FG-WSD scheduler in Nanbeige4-3B-Base, with corresponding data usage and learning rates.

Stage	Training Tokens	Learning Rate
Warmup Stage	0.1T	0 \rightarrow 4.5e-4
Diversity-Enriched Stable Stage	12.4T	Constant 4.5e-4
High-Quality Stable Stage	6.5T	Constant 4.5e-4
Decay Stage	4T	4.5e-4 \rightarrow 1.5e-6

Table 4: Comparison of base models on reasoning benchmarks after fine-tuning.

SFT Dataset	Base Model	AIME 2024	AIME 2025	Math-500	GPQA	LCB-V5	LCB-V6
Nemotron-Post-Training-V1	Qwen3-4B-Base	24.6	25.0	90.4	44.6	15.9	17.0
	Qwen3-8B-Base	37.9	29.6	91.1	48.9	27.6	28.1
	Nanbeige4-3B-Base	52.9	40.8	93.4	53.4	35.9	34.0
Ring-Lite-SFT	Qwen3-4B-Base	40.4	31.3	93.6	51.4	20.7	22.5
	Qwen3-8B-Base	50.0	35.8	94.4	55.1	30.2	29.5
	Nanbeige4-3B-Base	56.8	45.3	95.5	57.7	33.3	33.2
Openthoughts3	Qwen3-4B-Base	52.9	42.1	93.2	49.6	27.2	27.5
	Qwen3-8B-Base	60.4	47.1	95.0	55.3	35.2	34.4
	Nanbeige4-3B-Base	62.4	49.2	94.6	56.9	40.9	38.8

Stable, and **Decay**. The amount of training tokens and the learning rate schedule for each stage are detailed in Table 3. Notably, during the decay stage, we employ ABF (Adjusting Base Frequency) method [42] to extend the context length to 64K, allowing the model to fully assimilate synthetic long-CoTs, books, academic papers, and large-scale code repositories without truncation.

2.3 Post-SFT Evaluation for Base Model

Beyond evaluating base-model capabilities through few-shot evaluation, post-SFT downstream performance offers a more informative indicator, as virtually all production systems ultimately rely on instruction-tuned variants. We therefore fine-tune Nanbeige4-3B-Base and the open-source Qwen3-Base series with an identical supervised fine-tuning (SFT) pipeline and compared the resulting checkpoints on six representative benchmarks: AIME 2024, AIME 2025, GPQA [26], MATH-500 [18], LiveCodeBench-V5, and LiveCodeBench-V6 [14].

To ensure robustness, we repeat the procedure with three independent SFT datasets, including Nemotron-Post-Training-Dataset-V1 [21], Ring-Lite-SFT-Data [31], and OpenThoughts-3 [10], aggregating 9 SFT training runs in total. For each dataset, we randomly sample 500,000 instances and train for 2 epochs. To ensure a rigorously controlled and fair comparison, we keep all training hyperparameters identical across the different base models.

As shown in Table 4, across all the experiments, Nanbeige4-3B-Base not only surpasses Qwen3-4B-Base remarkably but also consistently outperforms the twice-as-large Qwen3-8B-Base, demonstrating that our model yields a stronger starting point for building downstream reasoning models.

3 Post-Training

In this section, we introduce each stage of the post-training process in detail. The overall post-training pipeline for Nanbeige4-3B-Thinking is illustrated in Figure 2. Initially, we conduct two fine-tuning stages—cold-start SFT and overall SFT—to equip the model with fundamental reasoning capabilities across a wide range of tasks. Subsequently, a knowledge-distillation phase is employed to further

enhance performance. Finally, the model undergoes multi-stage RL training to achieve additional improvements in multiple domain skills.

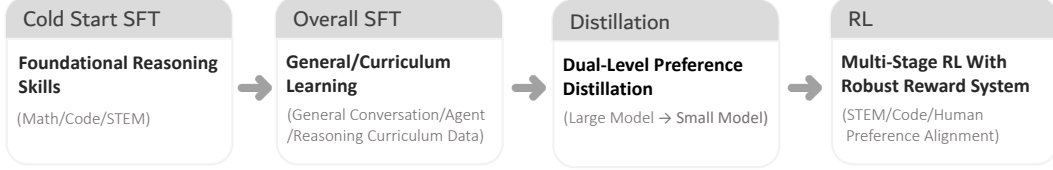


Figure 2: Nanbeige4-3B-Thinking Post-Training Pipeline

3.1 Cold Start Supervised Fine-tuning

On top of the base model, we first perform a cold-start supervised fine-tuning (SFT) stage to establish a robust foundation for reasoning. This stage focuses on high-quality reasoning data. After systematic cleaning and filtering, we collect approximately 30 million QA samples of mathematical, code, and subject-area problem-solving and reasoning. Based on this dataset, we construct a training corpus with a context length of 32K tokens, comprising approximately 50% mathematical reasoning, 30% scientific reasoning, and 20% code-related tasks. The objective of the cold-start stage is to strengthen the model’s chain-of-thought reasoning and structured response abilities [39, 36], providing a solid foundation for subsequent capability expansion.

Scaling SFT Instructions. While some recent technical reports suggest that tens or hundreds of thousands of high-quality instructions are sufficient for supervised-finetuning, our experiments on Nanbeige4-3B reveal a different trend. As shown in Figure 3, under comparable data distribution and quality, scaling the cold-start SFT instruction set from hundreds of thousands to tens of millions of examples leads to continued gains on challenging tasks like AIME 2025 and GPQA-Diamond, without a clear early saturation point. Motivated by this empirical observation, we explicitly adopt a Scaling SFT Instructions design: instead of relying on a compact instruction set, we train the model in the cold-start stage on tens of millions of carefully curated instructions, which yields a stronger reasoning prior and more stable chain-of-thought policy for later training stages.

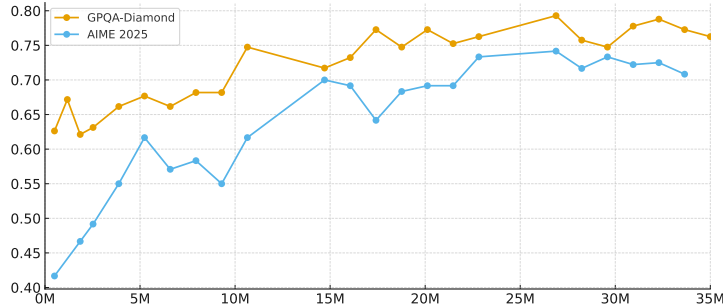


Figure 3: Cold-start SFT scaling (0.5M–35M) on NBG4-3B reasoning benchmarks.

3.2 Overall Supervised Fine-Tuning

After the model acquires initial reasoning capabilities, we perform an Overall SFT stage [22, 37, 48] to further enhance its general abilities and task diversity. The training corpus combines general conversation and writing data (covering everyday dialogue and multiple genres), agent-style interaction data (tool use, task decomposition, planning, and execution), harder reasoning data that targets the weak spots revealed in the cold-start stage, and code-related tasks that reinforce programming and code understanding. Using a 64K context length, we mix the data as roughly 40% mathematical and subject-specific reasoning, 30% general QA and writing, 20% agent scenarios, and 10% coding tasks, which systematically improves the model’s general dialogue ability, task execution, and adaptation

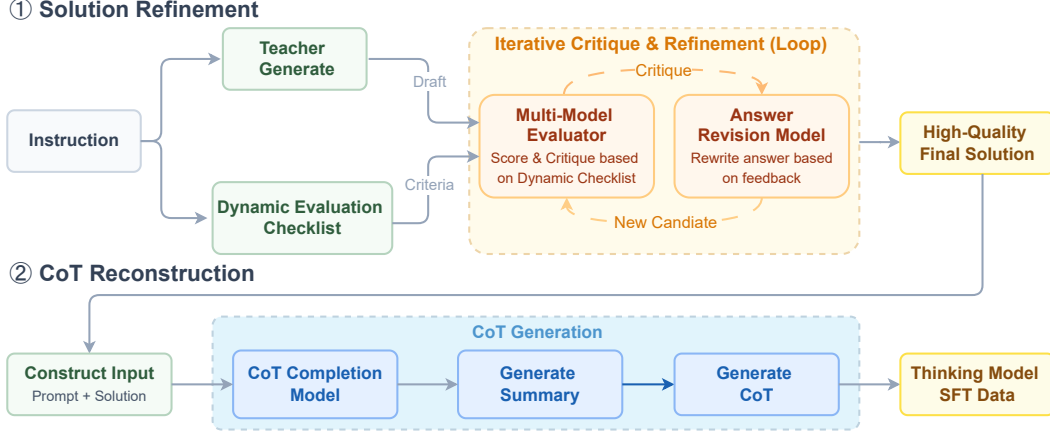


Figure 4: Deliberative generation refinement and CoT completion

to diverse application scenarios while maintaining strong reasoning performance. In the following section, we provide a detailed description of our data processing pipeline.

Solution Refinement. To enhance the model’s overall output quality on complex tasks, we develop a unified mechanism that combines deliberative generation refinement with chain completion [33]. Specifically, for each instruction, the system first constructs a tailored multi-dimensional evaluation checklist by selecting appropriate criteria from a predefined pool according to the instruction’s semantics and task type, and then adopts this checklist as an explicit constraint for the current evaluation round. The checklist selectively integrates criteria such as correctness, completeness, consistency, executability, and safety, and further refines each criterion into concrete checkpoints to enable fine-grained assessment of candidate responses. In the next iteration, the system dynamically selects from a pool of teacher models the one that performs best on the current instruction and has it co-generate candidate answers with the current SFT/Thinking model. An evaluation model then conducts cross-evaluation and comparative scoring of all candidates against a predefined checklist, producing structured feedback that highlights error locations, missing steps, and optimization suggestions. This feedback drives the Thinking model through iterative generate–review–revise cycles, continuously improving the quality of the solution.

CoT Reconstruction. After multiple rounds of deliberation and rewriting, although the final solution quality is greatly improved, the original chain of thought is often disrupted or lost, making it difficult to obtain supervision signals that simultaneously provide a high-quality final answer and a stable, learnable reasoning process. To address this, we additionally train a chain-completion model. When constructing training data for the Thinking model, we feed this model with the concatenation of the instruction and the final solution after multi-round refinement. Empirically, generating a summarized first improves followability, so the model first generates a brief summary chain of thought and then produces an explicit chain of thought that is consistent with the final answer. Finally, we concatenate the completed chain of thought and the final answer as the target output and use it as the training sample for the Thinking model, thus restoring a structured and well-aligned reasoning supervision signal while preserving answer quality.

After integrating the above mechanism into the Overall SFT stage, the model’s alignment with human preferences is significantly improved: on the Arena-Hard v2 benchmark, the score improves by 16%, with no degradation in reasoning capability [16].

Function Call Supporting. Our model provides native support for the function-call paradigm, enabling seamless tool invocation through formally defined and standardized parameter specifications. During the overall SFT stage, we proportionally increase the amount of function-call (FC) data to further strengthen this capability. To enhance the model’s proficiency in function calling, we incorporate both open-source and synthetic data. For the open-source portion, we unify data formats

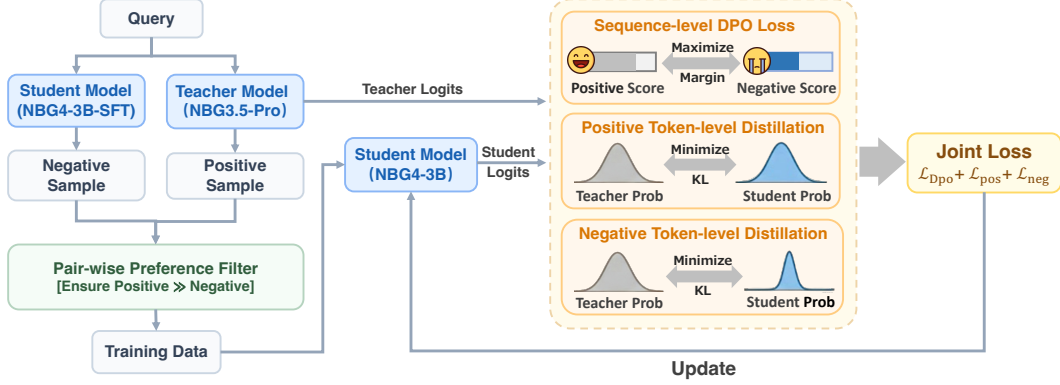


Figure 5: Overview of the Dual-Level Preference Distillation (DPD) framework

and reconstruct responses so that each sample strictly adheres to our function-call schema and includes an explicit reasoning path. For the synthetic portion, we deploy real-world environments and leverage strong models (e.g., Nanbeige3.5-Pro) to generate high-quality trajectories via rejection sampling. In addition, we adopt a multi-agent framework to simulate realistic user–assistant–tool interactions, creating data that span diverse scenarios and a wide range of difficulty levels [44].

3.3 Dual-level Preference Distillation

We propose **Dual-Level Preference Distillation (DPD)**, a joint training framework that harmonizes token-level knowledge distillation with sequence-level preference optimization. In this framework, Direct Preference Optimization (DPO) [24] acts as a sequence-level decision-boundary regularizer, maximizing the margin between positive and negative responses. Simultaneously, we introduce token-level supervision from the teacher model’s probability distribution on both sample types. This dual-granularity setup improves instruction-following behavior and subjective preference alignment, and at the same time, substantially enhances the model’s complex reasoning capabilities.

For data construction, positive samples are obtained by sampling multiple responses from the teacher model Nanbeige3.5-Pro for the same instruction, followed by model-based scoring and rule-based filtering to select the highest-scoring answer. Negative samples are generated by sampling from the 3B student model under training. These candidates are then passed through automatic evaluation and rule-based checks, and only those whose quality and scores are significantly worse than the corresponding positive sample are retained.

For the optimization objective, we use a joint loss that combines token-level probability distillation with a sequence-level DPO preference loss. On positive samples, the student model is trained to match the Nanbeige3.5 Pro probability distribution at each token. On negative samples, we also apply a distillation loss, where the teacher provides a reference distribution for the incorrect responses generated by the student, reducing the probability of highly confident erroneous tokens and increasing the probability of under-estimated but reasonable alternatives. This design enhances the model’s ability to correct its own mistakes and to recognize errors. The sequence level DPO preference loss, implemented as a margin constraint, explicitly enlarges the score gap between positive and negative responses, thereby sharpening the decision boundary and improving style alignment.

Using the Nanbeige4-3B SFT model as the baseline, training with this framework yields consistent and substantial relative gains on multiple internal and public benchmarks: around an 8% improvement on challenging mathematical benchmarks such as AIME 2024 and AIME 2025 [6], about a 10% gain on scientific reasoning tasks such as GPQA [25], and roughly a 30% improvement on BFCL V4, which emphasize tool use and problem decomposition [23]. At the same time, Arena-style subjective preference evaluations still show an improvement of around 8% [16]. Furthermore, incorporating an RL phase on top of this distillation framework yields substantially larger gains compared to initiating RL directly from the SFT baseline.

3.4 Reinforcement Learning

Large-scale supervised fine-tuning and distillation provide a strong foundation for the reasoning ability of NBG4-3B. We further bootstrap both its reasoning capability and human preference alignment through reinforcement learning (RL). Instead of conducting a single-stage RL procedure on a mixed corpus spanning multiple knowledge domains[43], we adopt a multi-stage RL framework, where each stage targets a specific ability dimension. To fully exploit the potential of each stage, we perform on-policy data filtering with the latest model before every RL phase. For each phase, we adopt appropriate reward models or verifiers based on the features of the training data.

3.4.1 On-Policy Data Filtering

During multi-stage RL training, the model’s reasoning ability improves after each stage. Due to cross-domain knowledge transfer, samples that were previously challenging may become trivial in later stages and contribute little to further improvement. To maintain a high-quality learning signal, we apply on-policy data filtering before each RL stage using the model from the previous stage.

Concretely, we use the model from the preceding stage to compute the avg@16 accuracy for every question and retain only those samples whose pass rate lies strictly between 10% and 90%. This focuses training on problems that are neither trivial nor unsolvable, thereby maximizing the effectiveness of each RL update.

3.4.2 Multi-Stage Reinforcement Learning

To enhance reasoning ability across domains, we employ multi-stage RL rather than a single mixed-corpus paradigm. While mixed-corpus training can yield strong cross-domain transfer[2], we observe that it often slows progress in specific domains: the model may spend many updates to achieve marginal gains in more challenging skills. For example, when jointly training on advanced mathematics and competitive programming data, the model tends to improve more on mathematics than on competitive coding. To address this imbalance, we organize RL into multiple stages, each focused on a single domain. This design allows the model to concentrate its capacity on domain-specific skills at each stage while still benefiting from cross-domain transfer over the full training pipeline.

We adopt on-policy GRPO [27] for each RL stage, with several enhancements for stabilized training. In particular, we remove the KL penalty term and mask the loss for truncated sequences, following the insights of DAPO [45]. We organize the training into three RL stages as follows.

STEM RL with Agentic Verifier. We use STEM-focused RL as the first stage, motivated by prior findings that STEM training provides strong cross-domain transfer. Our dataset consists of question-answer pairs in mathematics and the natural sciences. For mathematical problems, we collect diverse data from open-source datasets. For science problems spanning physics, chemistry, and biology, we use proprietary competition-level collections. For problems with multiple sub-parts, we rewrite each sub-problem into a self-contained question with a complete context.

In STEM domains, reference answers and model output may express equivalent numerical results in different symbolic forms. To provide accurate training signals, we employ a tool-augmented verifier that calls Python interpreter to perform exact computation and simplification [8]. This agentic verifier enables robust judgments that go beyond string-matching rules.

Practical Coding RL with Synthetic Test Functions. This stage aims to enhance practical coding ability across multiple programming languages and task scenarios. We design a multi-agent system to synthesize problems paired with executable test functions. To ensure the correctness and completeness of the synthetic data [3], we adopt a reverse-generation procedure: we first synthesize the solution and its corresponding test functions, and only then generate the natural-language problem description. Concretely, it first retrieves high-quality code snippets from GitHub, then refines or evolves these snippets into self-contained, verifiable solutions and produces paired public and private test functions. Finally, all candidate triples (problem, solution, test function) are validated via sandboxed execution to guarantee reliability. During RL training, these test functions are executed to provide a binary reward signal based on whether the generated solution passes all tests.

Table 5: Comparison between Nanbeige4-3B-Thinking and Qwen series reasoning models.

Benchmark	Qwen3-4B-2507	Qwen3-8B-2504	Qwen3-14B-2504	Qwen3-30A3-2507	Qwen3-32B-2504	Nanbeige4-3B-2511
Mathematical Reasoning						
AIME2025	81.3	67.3	70.4	<u>85.0</u>	72.9	85.6
AIME2024	83.3	76.0	79.3	<u>89.2</u>	81.4	90.4
Scientific Reasoning						
GPQA-Diamond	67.2	62.0	64.0	<u>73.4</u>	68.7	82.2
SuperGPQA	46.7	39.1	46.8	56.8	<u>54.1</u>	53.2
Tool Use & Coding						
BFCL-V4	44.9	42.2	45.4	<u>48.6</u>	47.9	53.8
Fullstack Bench	47.1	51.5	55.7	<u>54.4</u>	58.2	48.0
Human Preference Alignment						
ArenaHard-V2	40.5	26.4	39.9	60.0	48.4	60.0
Multi-Challenge	<u>41.8</u>	35.8	36.4	49.4	39.2	41.2

Human Preference Alignment RL with Pairwise Reward Model. In the final stage, we focus on aligning the model with human preferences on tasks such as creative writing and role-playing. Since these tasks prefer open-ended responses without fixed reference answers, many prior works rely on general-purpose language models to score the human preference alignment of candidate responses[17]. However, using a general language model as a reward model has two key drawbacks: (i) it often requires a lengthy chain of thought before reaching a final verdict, which is highly time-consuming; and (ii) RL training is prone to reward hacking. To address these issues, we train a pairwise reward model that can express preferences using only a few tokens, while exhibiting strong resistance to reward hacking.

During reinforcement learning (RL) training, we first sample diverse instructions that are both challenging and clearly specified. We then prompt strong baseline models (e.g., Nanbeige3.5-Pro) to generate high-quality reference responses. Each rollout produced by the policy model Nanbeige4-3B is paired with its corresponding reference response, and the pairwise reward model assigns a preference-based score that serves as the reward signal for policy optimization.

3.5 Post-Training Evaluation

We evaluate Nanbeige4-3B on a diverse suite of benchmarks to more comprehensively assess its performance. For each benchmark, we conduct multiple evaluation runs and report the average score across these repeated trials (e.g., avg@8 denotes the mean score computed over eight independent repetitions). Our evaluation encompasses the following benchmarks:

- **Mathematical Reasoning:** AIME 2025, AIME 2024[6] (reported as avg@8)
- **Scientific Reasoning:** GPQA-Diamond[26], superGPQA[5] (avg@3)
- **Tool Use & Coding:** BFCL-V4[23], Fullstack-Bench[1] (avg@3)
- **Human Preference Alignment:** Arena-Hard V2[17], Multi-Challenge[28] (avg@3)

For all benchmarks, we use a sampling temperature of 0.6 and top-p of 0.95, and we set the maximum generation length to 64k tokens. We compare Nanbeige4-3B against a series of open-source small language models: Qwen3-4B, Qwen3-8B, Qwen3-14B, Qwen3-30B-A3B, and Qwen3-32B. We report metrics of other models with recommended hyperparameters in our environment. Table 5 presents the evaluation results.

Experimental results show that Nanbeige4-3B excels at demanding reasoning tasks despite its compact size. It sets new state-of-the-art averages on AIME 2024, AIME 2025 and GPQA-Diamond, outperforming models with up to 10× more parameters, including Qwen3-32B and Qwen3-30B-A3B. Beyond mathematics and science, the model demonstrates strong tool-use proficiency, scoring 53.8 on BFCL-V4—an absolute gain of 5.2 points over Qwen3-30B-A3B. Human-preference alignment is

equally notable: Nanbeige4-3B matches the 60.0 point top score on Arena-Hard V2, yielding a 50% relative improvement over Qwen3-4B.

In addition to the benchmark results from our locally deployed evaluations, Nanbeige4-3B also demonstrates outstanding performance in external public evaluations. As shown in Table 1, Nanbeige4-3B-Thinking-2511 ranks among the top models on the WritingBench Leaderboard (November 2025), showcasing writing capabilities across diverse scenarios that are on par with much larger models.

4 Conclusion

In this work, we introduce Nanbeige4-3B, a compact yet highly capable 3-billion-parameter language model that redefines what small-scale models can achieve through innovation on data and training paradigm. Trained on a meticulously curated 23 trillion tokens of high-quality data and enhanced with novel post-training techniques—including chain-of-thought introspection and refinement, advanced reinforcement learning, and knowledge distillation—Nanbeige4-3B demonstrates remarkable reasoning and generation capabilities across diverse domains.

Despite its modest size, Nanbeige4-3B outperforms open-source counterparts such as Qwen3-8B and Qwen3-14B on challenging benchmarks, including AIME, superGPQA, Arena-Hard V2, and BFCL-V4. Notably, in the latest official WritingBench leaderboard, the Nanbeige4-3B-Thinking-2511 achieves top-tier performance, rivaling that of large-scale models like Deepseek-R1-0528. Nanbeige4-3B thus establishes a new lightweight flagship paradigm, offering high capability, efficiency, and accessibility for both research and real-world deployment.

Looking ahead, we aim to further extend the capabilities of small-scale models to even more complex challenges, such as autonomous software engineering (SWE), deep-research agents, and diverse real-world cross-scenario tool use tasks.

References

- [1] Bytedance-Seed-Foundation-Code-Team, Yao Cheng, Jianfeng Chen, Jie Chen, Li Chen, Liyu Chen, Wentao Chen, Zhengyu Chen, Shijie Geng, and et al. Fullstack bench: Evaluating llms as full stack coders, 2025.
- [2] Zhoujun Cheng, Shibo Hao, Tianyang Liu, Fan Zhou, Yutao Xie, Feng Yao, Yuexin Bian, Yonghao Zhuang, Nilabjo Dey, Yuheng Zha, Yi Gu, Kun Zhou, Yuqi Wang, Yuan Li, Richard Fan, Jianshu She, Chengqian Gao, Abulhair Saparov, Haonan Li, Taylor W. Killian, Mikhail Yurochkin, Zhengzhong Liu, Eric P. Xing, and Zhiting Hu. Revisiting reinforcement learning for llm reasoning from a cross-domain perspective, 2025.
- [3] Jason Chou, Ao Liu, Yuchi Deng, Zhiying Zeng, Tao Zhang, Haotian Zhu, Jianwei Cai, Yue Mao, Chenchen Zhang, Lingyun Tan, Ziyang Xu, Bohui Zhai, Hengyi Liu, Speed Zhu, Wiggan Zhou, and Fengzong Lian. Autocodebench: Large language models are automatic code benchmark generators, 2025.
- [4] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems, 2021.
- [5] Xinrun Du, Yifan Yao, Kaijing Ma, Bingli Wang, Tianyu Zheng, King Zhu, Minghao Liu, Yiming Liang, Xiaolong Jin, and et al. Supergpqa: Scaling llm evaluation across 285 graduate disciplines, 2025.
- [6] Emergent Mind. Aime 2024 and aime 2025 benchmarks. <https://www.emergentmind.com/topics/aime-2024-and-aime-2025-benchmarks>, 2025. Accessed 2025-11-29.
- [7] Zhaoye Fei, Yunfan Shao, Linyang Li, Zhiyuan Zeng, Hang Yan, Xipeng Qiu, and Dahua Lin. Query of CC: unearthing large scale domain-specific knowledge from public corpora. *CoRR*, abs/2401.14624, 2024.
- [8] Ruixiang Feng, Zhenwei An, Yuntao Wen, Ran Le, Yiming Jia, Chen Yang, Zongchao Chen, Lisi Chen, Shen Gao, Shuo Shang, Yang Song, and Tao Zhang. Cosineverifier: Tool-augmented answer verification for computation-oriented scientific questions, 2025.
- [9] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, and et al. The llama 3 herd of models, 2024.
- [10] Etash Guha, Ryan Marten, Sedrick Keh, Negin Raoof, Georgios Smyrnis, Hritik Bansal, Marianna Nezhurina, Jean Mercat, Trung Vu, and et al. Openthoughts: Data recipes for reasoning models, 2025.
- [11] Daya Guo, Dejian Yang, Haowei Zhang, Junxiao Song, Ruoyu Zhang, Runxin Xu, Qihao Zhu, Shirong Ma, Peiyi Wang, and et al. Deepseek-r1: Incentivizing reasoning capability in llms via reinforcement learning, 2025.
- [12] Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.
- [13] Shengding Hu, Yuge Tu, Xu Han, Chaoqun He, Ganqu Cui, Xiang Long, Zhi Zheng, Yewei Fang, Yuxiang Huang, and et al. Minicpm: Unveiling the potential of small language models with scalable training strategies, 2024.
- [14] Naman Jain, King Han, Alex Gu, Wen-Ding Li, Fanjia Yan, Tianjun Zhang, Sida Wang, Armando Solar-Lezama, Koushik Sen, and Ion Stoica. Livecodebench: Holistic and contamination free evaluation of large language models for code, 2024.
- [15] Haonan Li, Yixuan Zhang, Fajri Koto, Yifei Yang, Hai Zhao, Yeyun Gong, Nan Duan, and Timothy Baldwin. Cmmu: Measuring massive multitask language understanding in chinese, 2024.

- [16] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint*, 2024.
- [17] Tianle Li, Wei-Lin Chiang, Evan Frick, Lisa Dunlap, Tianhao Wu, Banghua Zhu, Joseph E. Gonzalez, and Ion Stoica. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024.
- [18] Hunter Lightman, Vineet Kosaraju, Yura Burda, Harri Edwards, Bowen Baker, Teddy Lee, Jan Leike, John Schulman, Ilya Sutskever, and Karl Cobbe. Let’s verify step by step, 2023.
- [19] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.
- [20] MiniMax, Aili Chen, Aonian Li, Bangwei Gong, Binyang Jiang, Bo Fei, Bo Yang, Boji Shan, Changqing Yu, Chao Wang, and et al. Minimax-m1: Scaling test-time compute efficiently with lightning attention, 2025.
- [21] Dhruv Nathawani, Igor Gitman, Somshubra Majumdar, Evelina Bakhturina, Ameya Sunil Mahabaleshwarkar, , Jian Zhang, and Jane Polak Scowcroft. Nemotron-Post-Training-Dataset-v1, 2025.
- [22] Long Ouyang, Jeff Wu, Xu Jiang, Diogo Almeida, et al. Training language models to follow instructions with human feedback. *Advances in Neural Information Processing Systems*, 2022.
- [23] Shishir G. Patil, Huanzhi Mao, Charlie Cheng-Jie Ji, Fanjia Yan, Vishnu Suresh, Ion Stoica, and Joseph E. Gonzalez. The berkeley function calling leaderboard (bfcl): From tool use to agentic evaluation of large language models. In *Forty-second International Conference on Machine Learning*, 2025.
- [24] Rafael Rafailov, Archit Sharma, Eric Mitchell, Stefano Ermon, Christopher D. Manning, and Chelsea Finn. Direct preference optimization: Your language model is secretly a reward model. *Advances in Neural Information Processing Systems*, 2024.
- [25] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark. *arXiv preprint*, 2023.
- [26] David Rein, Betty Li Hou, Asa Cooper Stickland, Jackson Petty, Richard Yuanzhe Pang, Julien Dirani, Julian Michael, and Samuel R. Bowman. Gpqa: A graduate-level google-proof q&a benchmark, 2023.
- [27] Zhihong Shao, Peiyi Wang, Qihao Zhu, Runxin Xu, Junxiao Song, Xiao Bi, Haowei Zhang, Mingchuan Zhang, Y. K. Li, Y. Wu, and Daya Guo. Deepseekmath: Pushing the limits of mathematical reasoning in open language models, 2024.
- [28] Ved Sirdeshmukh, Kaustubh Deshpande, Johannes Mols, Lifeng Jin, Ed-Yeremai Cardona, Dean Lee, Jeremy Kritz, Willow Primack, Summer Yue, and Chen Xing. Multichallenge: A realistic multi-turn conversation evaluation benchmark challenging to frontier llms, 2025.
- [29] Dan Su, Kezhi Kong, Ying Lin, Joseph Jennings, Brandon Norick, Markus Kliegl, Mostofa Patwary, Mohammad Shoeybi, and Bryan Catanzaro. Nemotron-cc: Transforming common crawl into a refined long-horizon pretraining dataset, 2025.
- [30] Mirac Suzgun, Nathan Scales, Nathanael Schärli, Sebastian Gehrmann, Yi Tay, Hyung Won Chung, Aakanksha Chowdhery, Quoc V. Le, Ed H. Chi, Denny Zhou, and Jason Wei. Challenging big-bench tasks and whether chain-of-thought can solve them, 2022.
- [31] Ling Team. Ring-lite: Scalable reasoning via c3po-stabilized reinforcement learning for llms, 2025.
- [32] Ling Team, Ang Li, Ben Liu, Binbin Hu, Bing Li, Bingwei Zeng, Borui Ye, Caizhi Tang, and Changxin Tian et al. Every activation boosted: Scaling general reasoner to 1 trillion open language foundation, 2025.

- [33] Tencent Hunyuan Team. Hunyuan-turbos: Advancing large language models through mamba-transformer synergy and adaptive chain-of-thought. *arXiv preprint*, 2025.
- [34] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, Aurelien Rodriguez, Armand Joulin, Edouard Grave, and Guillaume Lample. Llama: Open and efficient foundation language models, 2023.
- [35] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, and et al. Llama 2: Open foundation and fine-tuned chat models, 2023.
- [36] Xuezhi Wang, Jason Wei, Dale Schuurmans, Sharan Narang, Aakanksha Chowdhery, Ed H. Chi, Quoc V. Le, and Denny Zhou. Self-consistency improves chain of thought reasoning in language models. In *International Conference on Learning Representations*, 2023.
- [37] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A. Smith, Daniel Khashabi, and Hannaneh Hajishirzi. Self-instruct: Aligning language models with self-generated instructions. *Transactions of the Association for Computational Linguistics*, 2023.
- [38] Yubo Wang, Xueguang Ma, Ge Zhang, Yuansheng Ni, Abhranil Chandra, Shiguang Guo, Weiming Ren, Aaran Arulraj, Xuan He, Ziyang Jiang, Tianle Li, Max Ku, Kai Wang, Alex Zhuang, Rongqi Fan, Xiang Yue, and Wenhui Chen. Mmlu-pro: A more robust and challenging multi-task language understanding benchmark, 2024.
- [39] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H. Chi, Quoc V. Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. *Advances in Neural Information Processing Systems*, 2022.
- [40] Tianwen Wei, Jian Luan, Wei Liu, Shuang Dong, and Bin Wang. Cmath: Can your language model pass chinese elementary school math test?, 2023.
- [41] Yuning Wu, Jiahao Mei, Ming Yan, Chenliang Li, Shaopeng Lai, Yuran Ren, Zijia Wang, Ji Zhang, Mengyue Wu, Qin Jin, and Fei Huang. Writingbench: A comprehensive benchmark for generative writing, 2025.
- [42] Wenhan Xiong, Jingyu Liu, Igor Molybog, Hejia Zhang, Prajjwal Bhargava, Rui Hou, Louis Martin, Rashi Rungta, Karthik Abinav Sankararaman, Barlas Oguz, Madian Khabsa, Han Fang, Yashar Mehdad, Sharan Narang, Kshitiz Malik, Angela Fan, Shruti Bhosale, Sergey Edunov, Mike Lewis, Sinong Wang, and Hao Ma. Effective long-context scaling of foundation models, 2023.
- [43] An Yang, Anfeng Li, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chang Gao, Chengen Huang, Chenxu Lv, et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.
- [44] Chen Yang, Ran Le, Yun Xing, Zhenwei An, Zongchao Chen, Wayne Xin Zhao, Yang Song, and Tao Zhang. Toolmind technical report: A large-scale, reasoning-enhanced tool-use dataset. *arXiv preprint arXiv:2511.15718*, 2025.
- [45] Qiyang Yu, Zheng Zhang, Ruofei Zhu, Yufeng Yuan, Xiaochen Zuo, Yu Yue, Weinan Dai, Tiantian Fan, Gaohong Liu, Lingjun Liu, Xin Liu, Haibin Lin, Zhiqi Lin, Bole Ma, Guangming Sheng, Yuxuan Tong, Chi Zhang, Mofan Zhang, Wang Zhang, Hang Zhu, Jinhua Zhu, Jiaze Chen, Jiangjie Chen, Chengyi Wang, Hongli Yu, Yuxuan Song, Xiangpeng Wei, Hao Zhou, Jingjing Liu, Wei-Ying Ma, Ya-Qin Zhang, Lin Yan, Mu Qiao, Yonghui Wu, and Mingxuan Wang. Dapo: An open-source llm reinforcement learning system at scale, 2025.
- [46] Aohan Zeng, Xin Lv, Qinkai Zheng, Zhenyu Hou, Bin Chen, Chengxing Xie, Cunxiang Wang, Da Yin, Hao Zeng, and et al. GLM-4.5: agentic, reasoning, and coding (ARC) foundation models. *CoRR*, abs/2508.06471, 2025.
- [47] Ranchi Zhao, Zhen Leng Thai, Yifan Zhang, Shengding Hu, Yunqi Ba, Jie Zhou, Jie Cai, Zhiyuan Liu, and Maosong Sun. Decoratelm: Data engineering through corpus rating, tagging, and editing with language models, 2024.

- [48] Chunting Zhou, Pengfei Liu, Puxin Xu, Srinivasan Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, Lili Yu, et al. Lima: Less is more for alignment. *Advances in Neural Information Processing Systems*, 2023.