The Best of Both Worlds: Combining Parallel and Sequential Inference Scaling via Aggregation Fine-Tuning

Anonymous ACL submission

Abstract

Scaling data and model size has been proven effective for boosting the performance of large language models. In addition to training-time scaling, recent studies have revealed that increasing test-time computational resources can further improve performance. In this work, we introduce *Aggregation Fine-Tuning* (AFT), a supervised fine-tuning paradigm where the model learns to synthesize multiple draft responses, referred to as *proposals*, into a single, refined answer, termed *aggregation*. At inference time, we apply a *propose-and-aggregate* strategy that iteratively generates and aggregates proposals, effectively scaling inference-time computation *without* relying on external guidance such as a reward model. Empirical results across benchmark datasets demonstrate that AFT-trained models achieve substantial gains with test-time scaling, outperforming best-of-N baselines while eliminating the need for external reward signals. Notably, an AFT model, fine-tuned from Llama3.1-8B-Base with only 64k data, achieves a 41.3% LC win rate on AlpacaEval 2, surpassing significantly larger LLMs such as Llama3.1-405B-Instruct and GPT-4. By combining sequential refinement and parallel sampling, the propose-and-aggregate framework scales inference-time computation in a flexible manner.

1 Introduction

Large language models (LLMs) (Dubey et al., 2024; OpenAI, 2023) have demonstrated remarkable success in applications ranging from coding (Li et al., 2023a; Roziere et al., 2023) and reasoning (Yuan et al., 2023; Qin et al., 2024a) to AI assistance (Chiang et al., 2023), driven by the expansion of both data and model capacity during pre-training (Hoffmann et al., 2022; Kaplan et al., 2020) and supervised fine-tuning (SFT) (Yuan et al., 2023). Beyond training-time scaling, recent studies (Snell et al., 2024; Zhang et al., 2024b) indicate

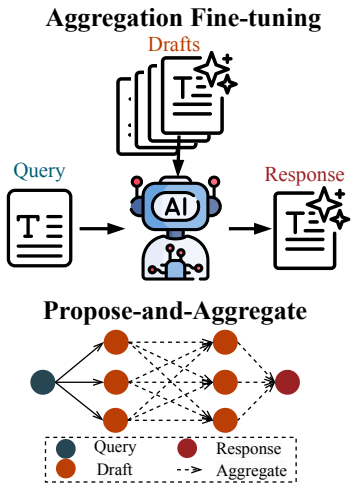

Figure 1: Aggregation Fine-Tuning: The model learns to generate a refined response by synthesizing multiple drafts conditioned on the query. Propose-and-aggregate: At inference time, the model recursively refines and aggregates draft responses to scale computation along parallel and sequential dimensions.

that allocating additional compute at inference can further boost performance, enabling smaller models to rival larger ones. Correspondingly, a variety of inference-time scaling algorithms (Madaan et al., 2023; Brown et al., 2024; Świechowski et al., 2023) have emerged, including selecting among multiple samples (*parallel sampling*) or sequentially revising generations (*sequential revision*).

Parallel sampling methods, such as Best-of-N (BoN), generate multiple candidate responses simultaneously, offering low latency but relying on an external verifier (e.g., a reward model) to identify the best output, which introduces additional system complexity. In contrast, sequential revision methods iteratively refine a single generation step by step. Although they do not require external verification, their inherently serial nature results in higher latency. This raises two key questions: (1) *Can we scale inference-time computation without relying on external verifiers?* (2) *Can we leverage the strengths of both parallel sampling and sequen-*

tial revision to unlock LLM capabilities for general tasks?

To address these questions, we introduce *Aggregation Fine-Tuning* (AFT), a new training paradigm that teaches a model to aggregate multiple draft responses into a single, refined output. AFT internalizes the aggregation process into the model itself, allowing it to learn how to evaluate, integrate, and refine multiple drafts during training. As illustrated in Figure 1, AFT replaces the conventional query-to-response mapping with a query–to–multi-draft–to–aggregation process, enabling the model to self-improve without external guidance. At inference time, the same mechanism naturally extends to a propose-and-aggregate strategy that recursively refines multiple drafts, as shown in Figure 1.

To perform aggregation fine-tuning, we begin by constructing a specialized training dataset in which each instance comprises a query, multiple draft responses (i.e., *proposals*), and a refined response (i.e., *aggregation*). Subsequently, the model is trained to aggregate proposals to an aggregation given the query, thereby extending conventional fine-tuning of query-to-response alignment. During inference, the model can perform *propose-and-aggregate* in a multi-layer fashion. At each layer, the model aggregates its proposals at the previous layer and generate proposals for the next layer. This iterative process of proposal generation and aggregation allows the model to leverage both parallel sampling and sequential revision effectively.

Empirical results on AlpacaEval 2 and MT-Bench demonstrate that AFT models achieve substantial performance gains under inference-time scaling while entirely eliminating the need for external guidance. Under comparable computational overhead, the propose-and-aggregate strategy significantly outperforms Best-of-N sampling, which depends on an external reward model for candidate selection. On AlpacaEval 2, it raises LC win rates above 30% for Mistral-based models and beyond 40% for Llama-based models, achieving improvements of up to +14.0% and +20.0%, respectively. Notably, the best-performing model, fine-tuned on only 64 k training examples and without any preference alignment, achieves an LC win rate of 41.3%, surpassing much larger models such as Llama-3.1-405B-Instruct and GPT-4. Further analyses reveal that both the diversity and quality of proposals substantially affect aggregation quality, and that conditioning generation on draft responses guides the model toward a lower-perplexity region, leading to faster and more stable convergence. Taken together, these results show that AFT effectively combines the strengths of sequential refinement and parallel sampling, offering a flexible and efficient paradigm for scaling inference-time computation.

2 Related Work

Supervised Fine-tuning of LLMs. Supervised fine-tuning plays a pivotal role in enhancing the instruction-following ability of LLMs (Li et al., 2023b; Zheng et al., 2023; Liu et al., 2024b; Ouyang et al., 2022a; Qin et al., 2024b; Jiang et al., 2024; Zhou et al., 2023b). Previous works find that high-quality instruction-following data is essential for the success of SFT (Zhao et al., 2024; Liu et al., 2024a; Xu et al., 2024; Zhou et al., 2023a; Li et al., 2024a) and various data-curation techniques are proposed (Wang et al., 2023; Du et al., 2023; Chen et al., 2024a; Li et al., 2024b; Sun et al., 2023). More specifically, including complex constrained (Sun et al., 2024; He et al., 2024), multi-faced (Lou et al., 2024), self-refined (Zelikman et al., 2022; Cheng et al., 2024; Wu et al., 2024a) and sequential (Hu et al., 2024) data could enhance the instruction-following ability of LLMs. Additionally, several studies on reasoning suggest incorporating intermediate processes (e.g., refinements in sequential revision (Cheng et al., 2024; Wu et al., 2024a) or search traces in step-level graph search (Qin et al., 2024a; Xi et al., 2024; Zhang et al., 2024a)) into the SFT data to enhance performance in reasoning tasks. The proposed aggregation fine-tuning differs in that it uses aggregation as the inference process of general-purpose instruction-following tasks to facilitate test-time scaling.

Inference Scaling for LLMs. Recent research has shifted attention from training-time computation to test-time scaling. Researchers show that allocating more compute at inference can sometimes prove more effective than enlarging model size (Snell et al., 2024). Indeed, with existing inference-time algorithms (Wei et al., 2022; Yao et al., 2023; Besta et al., 2024; Chen et al., 2024b; Brown et al., 2024), smaller models can surpass larger ones when both share a fixed test-time budget (Wu et al., 2024b). Previous work (Welleck et al., 2024; Snell et al., 2024) categorizes inference scaling into: *sequential revision*, where an LLM functions as a callable module to iteratively refine its prior outputs (Dohan et al., 2022;

Khattab et al., 2024; Madaan et al., 2023; Shinn et al., 2023; Welleck et al., 2023; Havrilla et al., 2024); *parallel sampling*, which generates multiple candidate responses concurrently and selects the best via a ranker (Brown et al., 2024; Huang et al., 2024; Jiang et al., 2023b); and *step-level graph search*, wherein a complex query is decomposed into intermediate steps explored through tree-search (Świechowski et al., 2023) or graph-search algorithms (Yao et al., 2023; Besta et al., 2024; Tian et al., 2024; Chi et al., 2024). While most of these methods focus on a single model's test-time scaling, researchers introduce a *mixture-of-agents* approach that integrates multiple LLMs and aggregates their diverse outputs to boost performance (Wang et al., 2024a). In this work, we internalize aggregation as a unified process within the model itself, where aggregation learning enables it to combine the complementary strengths of parallel and sequential inference, thereby scaling reasoning at test time without relying on any external verifier.

3 Method

In a traditional supervised fine-tuning setup, a language model $f_\theta : \mathcal{Q} \to \mathcal{R}$ is trained to map each query $q \in \mathcal{Q}$ directly to a single reference response $r^* \in \mathcal{R}$. In contrast, we propose a supervised training framework wherein a language model learns to aggregate multiple draft responses, referred to as *proposals*, into a refined final response, termed *aggregation*. We define this process as **aggregation fine-tuning**. Let $q \in \mathcal{Q}$ be a query, and $P = \{r_1, r_2, \ldots, r_K\} \subseteq \mathcal{R}$ represent a set of K proposals. Given (q, P), the goal of AFT is to train an aggregator $A_\theta : \mathcal{Q} \times \mathcal{R}^K \to \mathcal{R}$, parameterized by θ, to produce a refined response $\hat{r} = A_\theta(q, P)$ that approximates a high-quality reference response $r^* \in \mathcal{R}$. During inference, the model can *iteratively* perform aggregation $A_\theta : \mathcal{Q} \times \mathcal{R}_{t-1}^K \to \mathcal{R}_t$ to refine responses, where t is the iteration step or aggregation layer.

This process contrasts with traditional supervised fine-tuning, which directly maps a query q to a single reference response r^*. By explicitly conditioning on the drafts P, the model requires a meta-skill which goes beyond producing an output for a query. It learns a reasoning process that evaluates, critiques, and improves upon initial solutions. The overall framework is illustrated in Figure 2. We provide a detailed illustration of data construction, model training, and inference in the following sections.

3.1 Data Construction

Constructing training data for aggregation fine-tuning involves collecting model proposals and a reference aggregation response, for each user query.

Proposal Collection. We categorize proposals into two types: **off-policy** proposals and **on-policy** proposals. *Off-policy* proposals are derived from existing preference alignment datasets, such as UltraFeedback (Cui et al., 2023), where each query is accompanied by multiple responses generated by models different from the one being fine-tuned. *On-policy* proposals are obtained by leveraging in-context learning (ICL) with demonstrations, prompting the base LLM to generate multiple responses for a given query (Lin et al., 2024). Since these proposals are sampled from the same LLM that will be fine-tuned, they are referred to as *on-policy* proposals. A significant advantage of *on-policy* proposals is their reliance solely on the model being trained, i.e., they do not necessitate multiple policy models for proposal generation, thereby streamlining data construction. This approach enables the model to learn how to revise and aggregate its own outputs. Both methods result in a set of K proposals, denoted as $P = \{r_1, \ldots, r_K\}$, for each query q.

Aggregation Construction. We use a dedicated prompt (Appendix A) to prompt a stronger language model to aggregate the proposals P into a reference response r^* that serves as the ground truth for training. This procedure ensures that r^* demonstrates an ideal aggregation behavior, incorporating information from P while correcting errors and improving quality. The data construction details can be found in Section 4.

3.2 Training

The goal of aggregation fine-tuning is to maximize the probability that the model's aggregator A_θ assigns to the reference aggregated response r^*. Formally, for a training dataset $\{(q^{(i)}, P^{(i)}, r^{*(i)})\}_{i=1}^N$, where each example consists of a query $q^{(i)}$, a set of proposals $P^{(i)}$, and a high-quality reference aggregation $r^{*(i)}$, we seek:

$$\max_\theta \frac{1}{N} \sum_{i=1}^{N} A_\theta\big(r^{*(i)} \,|\, q^{(i)},\, P^{(i)}\big). \qquad (1)$$

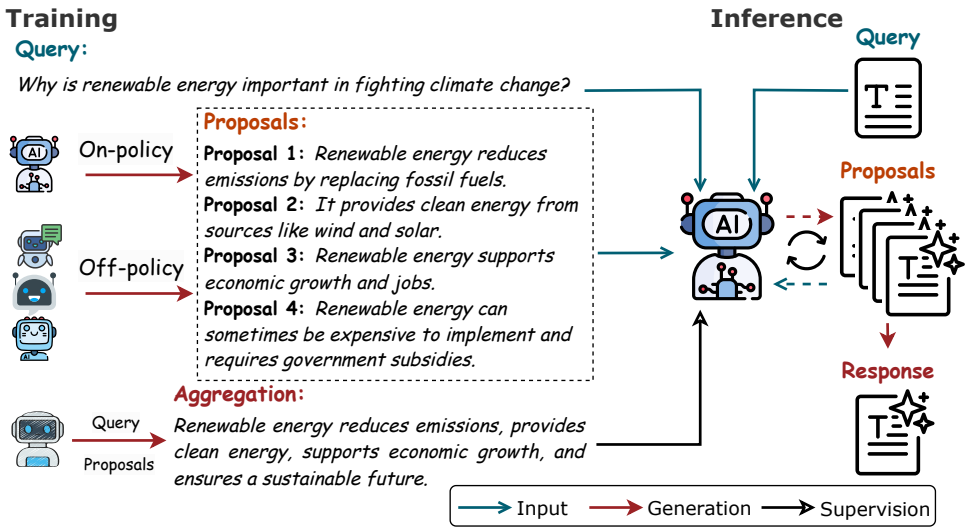

Figure 2: Framework of aggregation fine-tuning and propose-and-aggregate inference.

In contrast, conventional supervised training learns a mapping function f from a query to reference response directly:

$$\max_{\theta} \frac{1}{N} \sum_{i=1}^{N} f_\theta(r^{*(i)} \mid q^{(i)}) \tag{2}$$

We use the same prompt as outlined in Section 3.1 to incorporate proposals. Rather than producing an answer from scratch given only the query, the aggregator can build upon the semantic cues, linguistic structures, and partial solutions embedded in the drafts. As a result, the model's generation perplexity is significantly reduced, reflecting its ability to more efficiently predict the reference tokens due to the guidance offered by the existing candidates. Therefore, this enriched learning signal leads to a more stable and smoother training curve, as demonstrated in Section 6.3.

3.3 Inference

During inference, the AFT model can perform **propose-and-aggregate** to scale inference-time computing to boost performance. The process of propose-and-aggregate involves iteratively aggregating proposals from the previous layer and generating aggregations, forming the proposals for the next layer.

Initial Proposal Generation. Given a query q, the model draws K initial proposals $P^{(0)} = \{r_1^{(0)}, \ldots, r_K^{(0)}\}$ by sampling:

$$r_i^{(0)} \sim A_\theta(r|q) \quad i = 1, 2, \ldots K. \tag{3}$$

Choosing a single proposal from this set as the final response is equivalent to standard generation (i.e., $A_\theta(q)$) which does not scale inference-time computation.

Iterative Aggregation (Optional). Given the initial proposals, the model can optionally produce a refined response in a multi-layer setup. Specifically, the model can take in the aggregated responses from the previous layer ℓ (e.g., initial proposals at the first layer $P^{(0)}$) as input proposals:

$$P^{(\ell)} = \{ r_1^{(\ell)}, \ldots, r_K^{(\ell)} \} \quad \ell = 0, 1, \ldots, L, \tag{4}$$

where L is the maximum aggregation layer. Subsequently, the model performs aggregation on these proposals to further enhance generation quality:

$$r_i^{(\ell+1)} \sim A_\theta\big(r|q,\ P^{(\ell)}\big) \quad i = 1, 2, \ldots K, \tag{5}$$

forming the proposals for the next layer: $P^{(\ell+1)}$. This iterative procedure continues until a predefined number of layers is reached.

Iterative propose-and-aggregate inference thus emulates a recursive reasoning process in which the model refines its responses over multiple stages. In this way, propose-and-aggregate combines the depth-first nature of sequential revision with the breadth-first advantages of parallel sampling, but *does not* depend on an external verifier. Instead, the model itself refines and merges diverse aggregations from previous iterations.

4 Experimental Setup

Data. We construct training data based on UltraFeedback (Cui et al., 2023). UltraFeedback con-

sists of 64k instances, each of which containing multiple model responses to a user query, providing a natural testbed for *off-policy* proposals. Among these responses, we select the response with the highest score as the reference for training *SFT* baselines. To generate *on-policy* proposals, we utilize in-context learning with demonstrations to prompt base LLMs (e.g., Llama-3.1-8B-base and Mistral-7B-v0.1-Base) for sampling 10 responses to a query in UltraFeedback, subsequently employing a reward model to select the top 5 highest-ranked samples. We use Qwen2.5-72B-Instruct (Hui et al., 2024) to generate aggregated responses (aggregations) given queries and proposals following previous work (Wang et al., 2024a). To ablate the knowledge distillation effect, we also consider *SFT-KD* which fine-tunes the model using the aggregated responses.

Training. We perform fine-tuning on two families of base LLMs: Llama-3.1-8B-Base (Dubey et al., 2024) and Mistral-7B-v0.1-Base (Jiang et al., 2023a). All models' training is conducted using Llama-Factory (Zheng et al., 2024). All models are fine-tuned with LoRA (Hu et al., 2021) for three epochs, employing a rank of 8 and a total batch size of 128 across four A100 GPUs.

Evaluation. We primarily evaluate our models using two prominent open-ended instruction-following benchmarks: the single-turn dialogue evaluation AlpacaEval 2 (Li et al., 2023b) and the multi-turn dialogue evaluation MT-Bench (Zheng et al., 2023). Additionally, we also evaluate performance on downstream tasks which encompass general knowledge (MMLU, Hendrycks et al., 2021), reasoning abilities (ARC-c, Clark et al., 2018 and StrategyQA, Geva et al., 2021), mathematical skills (GSM8K, Cobbe et al., 2021), and instruction following (IFEval, Zhou et al., 2023b).

Inference. By default, we set the number of proposals to 5 and configure the aggregation layers to 2 for AlpacaEval 2 and MT-Bench, while employing a single-layer aggregation for GSM8K and IFEval. For beam search, we use a beam width of 21 for comparable computational overhead. For best-of-N, we adopt `sfairXC/FsfairX-LLaMA3-RM-v0.1` (Dong et al., 2023) as the external reward model and set the number of samples as 11 (Section 5.2). The more **detailed experimental settings** are presented in Appendix B.

5 Experimental Results

5.1 Main Results

Baselines. We evaluate AFT under two variants, i.e., **AFT-on-policy** and **AFT-off-policy**, and compare them against two standard fine-tuning baselines: (1) **SFT**, which learns from query-response pairs directly, and (2) **SFT-KD**, which is trained on aggregated responses without access to the underlying proposals. For test-time scaling, we consider two baselines: (1) **beam search**, which expands decoding *without* external supervision, and (2) **best-of-N (BoN)**, which uses an *external* reward model to select from multiple candidate responses (Section 5.2).

Direct Inference. Table 1 shows that both AFT variants and SFT-KD significantly outperform SFT across AlpacaEval 2 and MT-Bench, with AFT yielding larger improvements. Between the two AFT variants, *AFT-on-policy* consistently performs better than AFT-off-policy, likely due to its closer alignment with the model's own generation distribution. This allows the model to operate within a *low-perplexity region* that better matches its learned capabilities (see Section 6.3).

Inference Scaling. Since SFT methods cannot perform propose-and-aggregate inherently (Section 5.3), we adopt commonly used beam search to scale test-time compute for SFT-KD, and compare with propose-and-aggregate of AFT models. On AlpacaEval 2, propose-and-aggregate leads to substantial gains: LC win rates exceed 30.0% for Mistral-based models and surpass 40.0% for Llama-based ones, translating to improvements of up to +14.0% and +20.0%, respectively. The strongest result comes from the AFT-on-policy model fine-tuned from Llama3.1-8B-Base, achieving an LC win rate of **41.3%**, outperforming much larger models like GPT-4 and **Llama3.1-405B-Instruct**. By contrast, beam search not only fails to improve performance but can degrade it, likely due to post-SFT miscalibration (Zhu et al., 2023; He et al., 2023; Kong et al., 2020), where model confidence diverges from actual accuracy. Importantly, our AFT models are trained on just 64k instances and do not rely on preference optimization techniques such as DPO (Rafailov et al., 2023) or RLHF (Ouyang et al., 2022b). Despite being trained only on single-turn dialogues, AFT generalizes well to multi-turn tasks. In MT-Bench, propose-and-aggregate improves AFT-on-policy

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	MT-Bench 1st turn	MT-Bench 2nd turn	MT-Bench Average
GPT-3.5-Turbo (11/06)†	19.3	9.2	-	-	8.3
Mistral Medium†	28.6	21.9	-	-	8.6
GPT-4 (03/14)†	35.3	22.1	-	-	9.0
Llama 3.1-405B-Instruct†	39.3	39.1	-	-	-
Mistral-7B-v0.1-Base					
SFT	6.7	6.1	6.6	6.1	6.4
SFT-KD	18.8	22.6	7.5	6.1	6.8
w/ Beam Search	$14.2^{-4.6}$	$14.3^{-8.3}$	$7.4^{-0.1}$	$5.7^{-0.4}$	$6.6^{-0.2}$
AFT-off-policy	19.8	20.0	<u>7.7</u>	6.3	<u>7.0</u>
w/ Agg.	$\mathbf{33.8}^{+14.0}$	$47.8^{+27.8}$	$8.0^{+0.3}$	$7.0^{+0.7}$	$7.5^{+0.5}$
AFT-on-policy	<u>23.4</u>	<u>24.9</u>	7.5	<u>6.4</u>	6.9
w/ Agg.	$30.7^{+7.3}$	$\mathbf{48.4}^{+23.5}$	$\mathbf{8.3}^{+0.8}$	$\mathbf{7.0}^{+0.6}$	$\mathbf{7.6}^{+0.7}$
Llama3.1-8B-Base					
SFT	8.0	7.3	7.3	6.2	6.8
SFT-KD	18.4	<u>29.0</u>	<u>7.9</u>	6.7	7.3
w/ Beam Search	$19.9^{+1.5}$	$20.8^{-8.2}$	$7.5^{-0.4}$	$6.0^{-0.7}$	$6.8^{-0.5}$
AFT-off-policy	20.3	19.6	7.7	<u>6.9</u>	7.3
w/ Agg.	$40.3^{+20.0}$	$47.8^{+28.2}$	$8.3^{+0.6}$	$\mathbf{7.6}^{+0.7}$	$7.9^{+0.6}$
AFT-on-policy	<u>21.5</u>	21.8	<u>7.9</u>	<u>6.9</u>	<u>7.4</u>
w/ Agg.	$\mathbf{41.3}^{+19.8}$	$\mathbf{51.3}^{+29.5}$	$\mathbf{8.5}^{+0.6}$	$\mathbf{7.6}^{+0.7}$	$\mathbf{8.1}^{+0.7}$

Table 1: Single-turn and multi-turn dialogue evaluation on **AlpacaEval 2** and **MT-Bench**. † denotes results from the official leaderboard. Superscripts denote improvement over the preceding baseline with inference scaling. **Bold** and <u>underlined</u> numbers denote the best performance **with** and <u>without</u> inference scaling.

Model	AlpacaEval 2 LC(%)	AlpacaEval 2 WR(%)	FLOPs
SFT	8.0	7.3	F
w/ BoN (N=11)	15.7	13.5	$22F$
SFT-KD	18.4	29.0	F
w/ BoN (N=11)	29.2	43.7	$22F$
AFT-on-policy	21.5	21.8	F
w/ BoN (N=11)	32.7	35.4	$22F$
w/ Agg. (L=2,N=5)	**41.3**	**51.3**	$21F$

Table 2: Performance comparison between propose-and-aggregate and Best-of-N sampling (BoN) on models based on Llama-3.1-8B-Base.

Model	MMLU	ARC-c	StrategyQA
Mistral-7B-v0.1-Base			
SFT-KD	54.4	76.7	54.7
AFT-off-policy	**61.6**	80.0	70.1
AFT-on-policy	61.1	**80.7**	**70.9**
Llama3.1-8B-Base			
SFT-KD	60.1	77.6	64.1
AFT-off-policy	**63.1**	**79.6**	66.1
AFT-on-policy	61.8	78.6	**66.8**

Table 3: Downstream task performance (MMLU, HellaSwag and ARC-c).

scores from 6.9/7.4 to 7.6/8.1 for Mistral and Llama, respectively. A qualitative case study illustrating this improvement is provided in Appendix C.

5.2 Computational Overhead

We analyze the computational overhead of the propose-and-aggregate framework and compare it with parallel sampling. We approximate inference FLOPs following previous work (Brown et al., 2024) (Details in Appendix D). The primary additional computational cost arises from the aggregation step, which processes all proposals from the previous layer as input prompts. The FLOPs for propose-and-aggregate, denoted as $\hat{F}$, can be approximated as: $\hat{F} \approx L \cdot (2 \cdot N \cdot F) + F$, where F represents the FLOPs for vanilla generation, L is the number of aggregation layers and N denotes the number of parallel proposals.

Comparison with BoN. A common parallel sampling baseline is Best-of-N (BoN), which uses an external reward model to rank multiple generated responses. Assuming the reward model is the same size as the policy model, the FLOPs for BoN can be approximated by $\bar{F} \approx 2 \cdot N \cdot F$. Table 2 compares propose-and-aggregate to BoN under comparative FLOPs. While BoN consistently improves performance, it requires an external reward model for candidate selection. In contrast, propose-and-aggregate relies entirely on the model itself, i.e., **without** any external guidance, to improve upon its own proposals, and achieves significantly greater

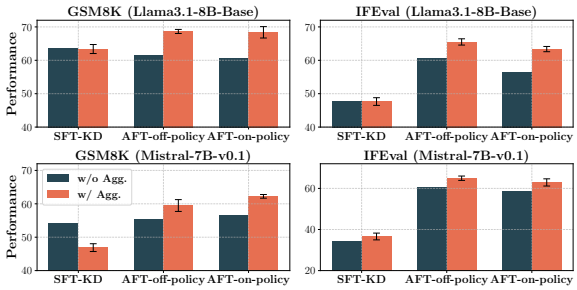

Figure 3: Performance of models fine-tuned on Llama-3.1-8B-Base on GSM8K and IFEval.

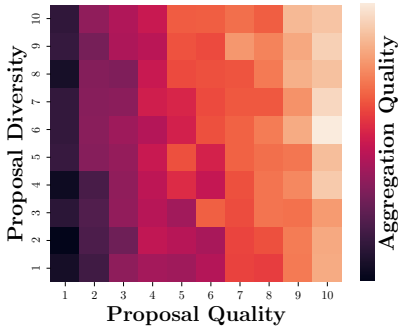

Figure 4: Aggregation quality in relation to proposal quality and diversity: each grid cell represents a diversity score (higher scores indicate better diversity). A lighter color indicates higher aggregation quality.

performance gains under similar computational budgets. These results highlight the advantage of scaling both along the width (parallel proposals) and depth (iterative aggregation) dimensions, rather than width alone as in BoN. We further analyze this advantage in Section 6.2.

5.3 Downstream Task Performance

As shown in Table 3, AFT models consistently outperform the SFT-KD baseline across diverse tasks, including general knowledge (MMLU) and reasoning benchmarks such as ARC-c and StrategyQA. Since the answers in these datasets are limited to short formats, e.g., single-choice or binary responses, propose-and-aggregate brings negligible improvement due to the limited informational content in proposals (Appendix F).

SFT cannot leverage propose-and-aggregate for test-time improvements. For downstream tasks like GSM8K and IFEval, we run propose-and-aggregate three times and report the mean and standard error. As shown in Figure 3, AFT models achieve competitive or superior performance compared to SFT-KD without inference-time scaling, and their advantages are further amplified when propose-and-aggregate is applied. In contrast, standard SFT-KD models lack aggregation capabilities and therefore *cannot* benefit from propose-and-aggregate at inference time.

We discuss the training overhead in Section 7 and present a FLOPs-fair comparison in Appendix E, showing that AFT consistently outperforms SFT-KD even with substantially fewer training examples.

6 Analysis

In this section, we delve into aggregation learning and the propose-and-aggregate framework. We begin by analyzing key proposal patterns that impact aggregation quality (Section 6.1), followed by experiments assessing test-time scaling in terms of search width and depth (Section 6.2). Subsequently, we investigate how aggregation learning outperforms traditional supervised fine-tuning (Section 6.3).

6.1 Effects of Proposal Diversity and Quality

We analyze how the diversity and quality of proposals affect aggregation performance. Using 100 instances sampled from AlpacaEval 2, we employ a reward model to evaluate responses generated by the AFT-on-policy model (Llama3.1-8B-Base). For each query, 10 proposals are sampled and all $\binom{10}{5} = 25{,}200$ subsets of five proposals are traversed. The model generates aggregations for each subset, whose quality is then scored by the same reward model. Proposal diversity is measured using the Vendi score (Friedman and Dieng, 2023), and proposal quality is defined as the average reward score of the subset. Scores are normalized per query on a 1-10 ranking scale, and the averaged results are shown in Figure 4. Both proposal diversity and quality substantially influence aggregation performance, with quality exerting a stronger effect, indicated by larger variations along the horizontal axis.

To further examine whether aggregation enables higher-order reasoning beyond simply selecting the best proposal, we compare each aggregation with the best individual proposal in its subset. Across 25,200 subsets, the average reward scores are 0.0041 for a random proposal, 1.27 for the best proposal, and 1.37 for the aggregation, with the aggregated response surpassing the best proposal in 61.4% of cases. These results suggest that AFT-trained models do more than internally mimic a reward model: they synthesize complementary in-

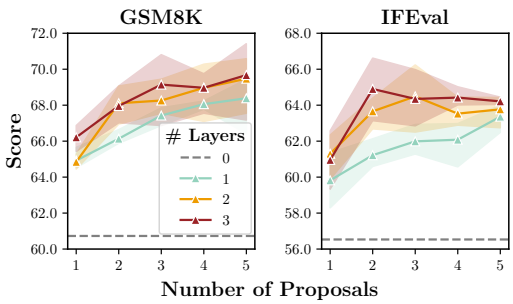

Figure 5: Performance w.r.t. aggregation width (number of proposals per aggregation layer) and depth (number of aggregation layers).

formation across drafts to generate responses that often exceed the quality of any single input, reflecting a capacity for higher-order reasoning.

6.2 Test-time Scaling along Width and Depth

As discussed in Section 3.3, the propose-and-aggregate framework unifies the strengths of sequential revision and parallel sampling to improve inference performance. To assess its scalability, we evaluate AFT-on-policy (Llama3.1-8B-Base) on GSM8K and IFEval, averaging results over three runs. Figure 4 shows that increasing both the search width (number of proposals) and depth (number of aggregation layers) generally leads to consistent performance gains. Interestingly, propose-and-aggregate reduces to sequential revision when using a single proposal, and to parallel sampling when using only one aggregation layer, where the model effectively serves as its own verifier. These findings highlight the framework's flexibility in adapting to different computational and latency constraints: devices with ample memory can scale along width, while memory-constrained settings may prefer greater depth at the cost of increased latency.

6.3 Understanding Aggregation Learning

The training curves in the left panel of Figure 6 show that aggregation fine-tuning is both more efficient and stable than standard SFT, achieving lower loss, faster convergence, and smoother progression. This indicates that AFT introduces smaller perturbations to the base model's distribution.

To further analyze this behavior, we sample 1,000 instances from each training dataset and compute the base model's perplexity before fine-tuning, as shown in the right panel of Figure 6. Perplexity on AFT data is markedly lower than on SFT data, suggesting that aggregation learning acts as a form of "mode seeking," concentrating probability mass on high-reward responses (Tajwar et al., 2024). Predicting a final answer after reviewing multiple drafts is less uncertain than generating one from scratch, leading to lower perplexity and faster convergence, especially in the on-policy setting, where drafts originate from the same model.

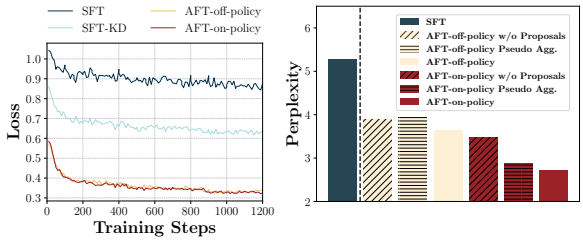

Figure 6: **Left**: training curves of SFT and AFT models (Llama3.1-8B-Base); **Right**: the perplexity of the base LLM of different training sets *before* fine-tuning, where "w/o Proposals" indicates removing proposals when calculating perplexity, and "Pseudo Agg." denotes using a pseudo aggregation in replacement of the real aggregation aggregated from proposals.

We also examine two ablation variants: (1) training on aggregations without proposals (SFT-KD), and (2) using direct responses from Qwen2.5-72B-Instruct as pseudo aggregations. The first variant resembles knowledge distillation from a stronger model, while the second disregards any underlying connection between proposals and aggregation, resulting in higher perplexity and less stable training, as shown in Figure 6. These findings highlight that explicitly learning to aggregate proposals, rather than distilling from pre-aggregated data, is key to AFT's efficiency and stability.

7 Conclusion

In this work, we introduced Aggregation Fine-Tuning (AFT), a training paradigm that enables language models to synthesize multiple draft responses into a refined answer. Beyond training, our propose-and-aggregate mechanism serves as a powerful form of test-time scaling, allowing models to recursively refine and integrate multiple generations without any external verifier. This unified process internalizes both parallel exploration and sequential refinement, scaling inference-time computation along width and depth simultaneously. Through this approach, AFT transforms test-time computation from a passive decoding process into an active, self-improving reasoning loop. These results establish propose-and-aggregate as an efficient and flexible pathway for unlocking the potential of LLMs through scalable test-time scaling.

Limitations

This work presents two main limitations. First, the construction of AFT training data relies on access to a stronger language model to generate high-quality aggregation targets. This requirement may introduce additional cost or dependency on proprietary systems, which could limit accessibility in some settings. Second, AFT training itself incurs higher computational overhead than standard SFT, as the aggregation stage increases the effective training FLOPs by roughly $(N + 1)\times$. Although the overall cost remains manageable, it can be non-trivial for large-scale training. Third, the propose-and-aggregate framework incurs additional computational overhead at inference time, as it involves generating and aggregating multiple draft responses. While this strategy yields notable performance gains, it may increase latency and resource consumption, which should be considered in deployment scenarios with strict efficiency constraints.

Ethics Considerations

Our work focuses on improving model training and inference efficiency through supervised aggregation, without introducing new data or user interaction components. All experiments are conducted using publicly available datasets and models (e.g., Llama3, AlpacaEval) under their original licenses, and we comply with their respective usage terms. AFT does not amplify known ethical or societal risks beyond those inherent to large language models. No human data collection, personal information, or sensitive content is involved. The proposed aggregation mechanism aims to improve reasoning quality and computational flexibility, and is intended solely for research and reproducibility purposes. AI tools are used only for language refinement of the manuscript, not for content generation or analysis.

References

Maciej Besta, Nils Blach, Ales Kubicek, Robert Gerstenberger, Michal Podstawski, Lukas Gianinazzi, Joanna Gajda, Tomasz Lehmann, Hubert Niewiadomski, Piotr Nyczyk, and 1 others. 2024. Graph of thoughts: Solving elaborate problems with large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 17682–17690.

Bradley C. A. Brown, Jordan Juravsky, Ryan Saul Ehrlich, Ronald Clark, Quoc V. Le, Christopher Ré, and Azalia Mirhoseini. 2024. Large language monkeys: Scaling inference compute with repeated sampling. *CoRR*, abs/2407.21787.

Lichang Chen, Shiyang Li, Jun Yan, Hai Wang, Kalpa Gunaratna, Vikas Yadav, Zheng Tang, Vijay Srinivasan, Tianyi Zhou, Heng Huang, and Hongxia Jin. 2024a. Alpagasus: Training a better alpaca with fewer data. In *The Twelfth International Conference on Learning Representations*.

Xinyun Chen, Renat Aksitov, Uri Alon, Jie Ren, Kefan Xiao, Pengcheng Yin, Sushant Prakash, Charles Sutton, Xuezhi Wang, and Denny Zhou. 2024b. Universal self-consistency for large language models. In *ICML 2024 Workshop on In-Context Learning*.

Jiale Cheng, Xiao Liu, Cunxiang Wang, Xiaotao Gu, Yida Lu, Dan Zhang, Yuxiao Dong, Jie Tang, Hongning Wang, and Minlie Huang. 2024. Spar: Self-play with tree-search refinement to improve instruction-following in large language models. *arXiv preprint arXiv:2412.11605*.

Yizhou Chi, Kevin Yang, and Dan Klein. 2024. Thoughtsculpt: Reasoning with intermediate revision and search. *arXiv preprint arXiv:2404.05966*.

Wei-Lin Chiang, Zhuohan Li, Zi Lin, Ying Sheng, Zhanghao Wu, Hao Zhang, Lianmin Zheng, Siyuan Zhuang, Yonghao Zhuang, Joseph E. Gonzalez, Ion Stoica, and Eric P. Xing. 2023. Vicuna: An open-source chatbot impressing gpt-4 with 90%* chatgpt quality.

Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. 2018. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457.

Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Mark Chen, Heewoo Jun, Lukasz Kaiser, Matthias Plappert, Jerry Tworek, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. 2021. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*.

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

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

David Dohan, Winnie Xu, Aitor Lewkowycz, Jacob Austin, David Bieber, Raphael Gontijo Lopes, Yuhuai Wu, Henryk Michalewski, Rif A Saurous, Jascha Sohl-Dickstein, and 1 others. 2022. Language model cascades. *arXiv preprint arXiv:2207.10342*.

Hanze Dong, Wei Xiong, Deepanshu Goyal, Rui Pan, Shizhe Diao, Jipeng Zhang, Kashun Shum, and Tong Zhang. 2023. Raft: Reward ranked finetuning for generative foundation model alignment. *arXiv preprint arXiv:2304.06767*.

Qianlong Du, Chengqing Zong, and Jiajun Zhang. 2023. Mods: Model-oriented data selection for instruction tuning. *Preprint*, arXiv:2311.15653.

Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, and et al. 2024. The llama 3 herd of models. *CoRR*, abs/2407.21783.

Yann Dubois, Balázs Galambosi, Percy Liang, and Tatsunori B Hashimoto. 2024. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*.

Dan Friedman and Adji Bousso Dieng. 2023. The vendi score: A diversity evaluation metric for machine learning. *Trans. Mach. Learn. Res.*, 2023.

Mor Geva, Daniel Khashabi, Elad Segal, Tushar Khot, Dan Roth, and Jonathan Berant. 2021. Did aristotle use a laptop? A question answering benchmark with implicit reasoning strategies. *Trans. Assoc. Comput. Linguistics*, 9:346–361.

Alexander Havrilla, Sharath Chandra Raparthy, Christoforos Nalmpantis, Jane Dwivedi-Yu, Maksym Zhuravinskyi, Eric Hambro, and Roberta Raileanu. 2024. GLore: When, where, and how to improve LLM reasoning via global and local refinements. In *Forty-first International Conference on Machine Learning*.

Guande He, Peng Cui, Jianfei Chen, Wenbo Hu, and Jun Zhu. 2023. Investigating uncertainty calibration of aligned language models under the multiple-choice setting. *CoRR*, abs/2310.11732.

Qianyu He, Jie Zeng, Qianxi He, Jiaqing Liang, and Yanghua Xiao. 2024. From complex to simple: Enhancing multi-constraint complex instruction following ability of large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2024*, pages 10864–10882, Miami, Florida, USA. Association for Computational Linguistics.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. 2021. Measuring massive multitask language understanding. In *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*. OpenReview.net.

Jordan Hoffmann, Sebastian Borgeaud, Arthur Mensch, Elena Buchatskaya, Trevor Cai, Eliza Rutherford, Diego de Las Casas, Lisa Anne Hendricks, Johannes Welbl, Aidan Clark, and 1 others. 2022. Training compute-optimal large language models. *arXiv preprint arXiv:2203.15556*.

Edward J. Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, and Weizhu Chen. 2021. Lora: Low-rank adaptation of large language models. *Preprint*, arXiv:2106.09685.

Hanxu Hu, Simon Yu, Pinzhen Chen, and Edoardo M Ponti. 2024. Fine-tuning large language models with sequential instructions. *arXiv preprint arXiv:2403.07794*.

Baizhou Huang, Shuai Lu, Xiaojun Wan, and Nan Duan. 2024. Enhancing large language models in coding through multi-perspective self-consistency. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1429–1450, Bangkok, Thailand. Association for Computational Linguistics.

Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Keming Lu, Kai Dang, Yang Fan, Yichang Zhang, An Yang, Rui Men, Fei Huang, Bo Zheng, Yibo Miao, Shanghaoran Quan, and 5 others. 2024. Qwen2.5-coder technical report. *Preprint*, arXiv:2409.12186.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. 2023a. Mistral 7b. *Preprint*, arXiv:2310.06825.

Dongfu Jiang, Xiang Ren, and Bill Yuchen Lin. 2023b. LLM-blender: Ensembling large language models with pairwise ranking and generative fusion. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14165–14178, Toronto, Canada. Association for Computational Linguistics.

Yuxin Jiang, Yufei Wang, Xingshan Zeng, Wanjun Zhong, Liangyou Li, Fei Mi, Lifeng Shang, Xin Jiang, Qun Liu, and Wei Wang. 2024. FollowBench: A multi-level fine-grained constraints following benchmark for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4667–4688, Bangkok, Thailand. Association for Computational Linguistics.

Jared Kaplan, Sam McCandlish, Tom Henighan, Tom B. Brown, Benjamin Chess, Rewon Child, Scott Gray, Alec Radford, Jeffrey Wu, and Dario Amodei. 2020. Scaling laws for neural language models. *CoRR*, abs/2001.08361.

Omar Khattab, Arnav Singhvi, Paridhi Maheshwari, Zhiyuan Zhang, Keshav Santhanam, Sri Vardhamanan A, Saiful Haq, Ashutosh Sharma, Thomas T. Joshi, Hanna Moazam, Heather Miller, Matei Zaharia, and Christopher Potts. 2024. DSPy: Compiling declarative language model calls into state-

of-the-art pipelines. In *The Twelfth International Conference on Learning Representations*.

Lingkai Kong, Haoming Jiang, Yuchen Zhuang, Jie Lyu, Tuo Zhao, and Chao Zhang. 2020. Calibrated language model fine-tuning for in- and out-of-distribution data. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 1326–1340. Association for Computational Linguistics.

Ming Li, Yong Zhang, Zhitao Li, Jiuhai Chen, Lichang Chen, Ning Cheng, Jianzong Wang, Tianyi Zhou, and Jing Xiao. 2024a. From quantity to quality: Boosting LLM performance with self-guided data selection for instruction tuning. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers)*, pages 7602–7635, Mexico City, Mexico. Association for Computational Linguistics.

Raymond Li, Loubna Ben Allal, Yangtian Zi, Niklas Muennighoff, Denis Kocetkov, Chenghao Mou, Marc Marone, Christopher Akiki, Jia Li, Jenny Chim, and 1 others. 2023a. Starcoder: may the source be with you! *arXiv preprint arXiv:2305.06161*.

Xuechen Li, Tianyi Zhang, Yann Dubois, Rohan Taori, Ishaan Gulrajani, Carlos Guestrin, Percy Liang, and Tatsunori B. Hashimoto. 2023b. Alpacaeval: An automatic evaluator of instruction-following models. `https://github.com/tatsu-lab/alpaca_eval`.

Yunshui Li, Binyuan Hui, Xiaobo Xia, Jiaxi Yang, Min Yang, Lei Zhang, Shuzheng Si, Ling-Hao Chen, Junhao Liu, Tongliang Liu, Fei Huang, and Yongbin Li. 2024b. One-shot learning as instruction data prospector for large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4586–4601, Bangkok, Thailand. Association for Computational Linguistics.

Bill Yuchen Lin, Abhilasha Ravichander, Ximing Lu, Nouha Dziri, Melanie Sclar, Khyathi Raghavi Chandu, Chandra Bhagavatula, and Yejin Choi. 2024. The unlocking spell on base llms: Rethinking alignment via in-context learning. In *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net.

Wei Liu, Weihao Zeng, Keqing He, Yong Jiang, and Junxian He. 2024a. What makes good data for alignment? a comprehensive study of automatic data selection in instruction tuning. In *The Twelfth International Conference on Learning Representations*.

Xiao Liu, Xuanyu Lei, Shengyuan Wang, Yue Huang, Andrew Feng, Bosi Wen, Jiale Cheng, Pei Ke, Yifan Xu, Weng Lam Tam, Xiaohan Zhang, Lichao Sun, Xiaotao Gu, Hongning Wang, Jing Zhang, Minlie Huang, Yuxiao Dong, and Jie Tang. 2024b. AlignBench: Benchmarking Chinese alignment of large language models. In *Proceedings of the 62nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 11621–11640, Bangkok, Thailand. Association for Computational Linguistics.

Renze Lou, Kai Zhang, Jian Xie, Yuxuan Sun, Janice Ahn, Hanzi Xu, Yu Su, and Wenpeng Yin. 2024. MUFFIN: Curating multi-faceted instructions for improving instruction following. In *The Twelfth International Conference on Learning Representations*.

Aman Madaan, Niket Tandon, Prakhar Gupta, Skyler Hallinan, Luyu Gao, Sarah Wiegreffe, Uri Alon, Nouha Dziri, Shrimai Prabhumoye, Yiming Yang, Shashank Gupta, Bodhisattwa Prasad Majumder, Katherine Hermann, Sean Welleck, Amir Yazdanbakhsh, and Peter Clark. 2023. Self-refine: Iterative refinement with self-feedback. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Yu Meng, Mengzhou Xia, and Danqi Chen. 2024. Simpo: Simple preference optimization with a reference-free reward. *CoRR*, abs/2405.14734.

OpenAI. 2023. GPT-4 technical report. *CoRR*, abs/2303.08774.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, and 1 others. 2022a. Training language models to follow instructions with human feedback. *Advances in neural information processing systems*, 35:27730–27744.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll L. Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F. Christiano, Jan Leike, and Ryan Lowe. 2022b. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems 35: Annual Conference on Neural Information Processing Systems 2022, NeurIPS 2022, New Orleans, LA, USA, November 28 - December 9, 2022*.

Yiwei Qin, Xuefeng Li, Haoyang Zou, Yixiu Liu, Shijie Xia, Zhen Huang, Yixin Ye, Weizhe Yuan, Hector Liu, Yuanzhi Li, and 1 others. 2024a. O1 replication journey: A strategic progress report–part 1. *arXiv preprint arXiv:2410.18982*.

Yiwei Qin, Kaiqiang Song, Yebowen Hu, Wenlin Yao, Sangwoo Cho, Xiaoyang Wang, Xuansheng Wu, Fei Liu, Pengfei Liu, and Dong Yu. 2024b. Infobench: Evaluating instruction following ability in large language models. *arXiv preprint arXiv:2401.03601*.

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

Baptiste Roziere, Jonas Gehring, Fabian Gloeckle, Sten Sootla, Itai Gat, Xiaoqing Ellen Tan, Yossi Adi, Jingyu Liu, Romain Sauvestre, Tal Remez, and 1 others. 2023. Code llama: Open foundation models for code. *arXiv preprint arXiv:2308.12950*.

Noah Shinn, Federico Cassano, Ashwin Gopinath, Karthik R Narasimhan, and Shunyu Yao. 2023. Reflexion: language agents with verbal reinforcement learning. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Charlie Snell, Jaehoon Lee, Kelvin Xu, and Aviral Kumar. 2024. Scaling LLM test-time compute optimally can be more effective than scaling model parameters. *CoRR*, abs/2408.03314.

Haoran Sun, Lixin Liu, Junjie Li, Fengyu Wang, Baohua Dong, Ran Lin, and Ruohui Huang. 2024. Conifer: Improving complex constrained instruction-following ability of large language models. *arxiv preprint arXiv:2404.02823*.

Zhiqing Sun, Yikang Shen, Qinhong Zhou, Hongxin Zhang, Zhenfang Chen, David Daniel Cox, Yiming Yang, and Chuang Gan. 2023. Principle-driven self-alignment of language models from scratch with minimal human supervision. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Maciej Świechowski, Konrad Godlewski, Bartosz Sawicki, and Jacek Mańdziuk. 2023. Monte carlo tree search: A review of recent modifications and applications. *Artificial Intelligence Review*, 56(3):2497–2562.

Fahim Tajwar, Anikait Singh, Archit Sharma, Rafael Rafailov, Jeff Schneider, Tengyang Xie, Stefano Ermon, Chelsea Finn, and Aviral Kumar. 2024. Preference fine-tuning of llms should leverage suboptimal, on-policy data. In *Forty-first International Conference on Machine Learning, ICML 2024, Vienna, Austria, July 21-27, 2024*. OpenReview.net.

Ye Tian, Baolin Peng, Linfeng Song, Lifeng Jin, Dian Yu, Lei Han, Haitao Mi, and Dong Yu. 2024. Toward self-improvement of LLMs via imagination, searching, and criticizing. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Junlin Wang, Jue Wang, Ben Athiwaratkun, Ce Zhang, and James Zou. 2024a. Mixture-of-agents enhances large language model capabilities. *CoRR*, abs/2406.04692.

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

Zhilin Wang, Yi Dong, Jiaqi Zeng, Virginia Adams, Makesh Narsimhan Sreedhar, Daniel Egert, Olivier Delalleau, Jane Polak Scowcroft, Neel Kant, Aidan Swope, and Oleksii Kuchaiev. 2024b. Helpsteer: Multi-attribute helpfulness dataset for steerlm. In *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 1: Long Papers), NAACL 2024, Mexico City, Mexico, June 16-21, 2024*, pages 3371–3384. Association for Computational Linguistics.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed H. Chi, Quoc V Le, and Denny Zhou. 2022. Chain of thought prompting elicits reasoning in large language models. In *Advances in Neural Information Processing Systems*.

Sean Welleck, Amanda Bertsch, Matthew Finlayson, Hailey Schoelkopf, Alex Xie, Graham Neubig, Ilia Kulikov, and Zaid Harchaoui. 2024. From decoding to meta-generation: Inference-time algorithms for large language models. *arXiv preprint arXiv:2406.16838*.

Sean Welleck, Ximing Lu, Peter West, Faeze Brahman, Tianxiao Shen, Daniel Khashabi, and Yejin Choi. 2023. Generating sequences by learning to self-correct. In *The Eleventh International Conference on Learning Representations*.

Tianhao Wu, Weizhe Yuan, Olga Golovneva, Jing Xu, Yuandong Tian, Jiantao Jiao, Jason Weston, and Sainbayar Sukhbaatar. 2024a. Meta-rewarding language models: Self-improving alignment with llm-as-a-meta-judge. *arXiv preprint arXiv:2407.19594*.

Yangzhen Wu, Zhiqing Sun, Shanda Li, Sean Welleck, and Yiming Yang. 2024b. An empirical analysis of compute-optimal inference for problem-solving with language models.

Zhiheng Xi, Dingwen Yang, Jixuan Huang, Jiafu Tang, Guanyu Li, Yiwen Ding, Wei He, Boyang Hong, Shihan Do, Wenyu Zhan, and 1 others. 2024. Enhancing llm reasoning via critique models with test-time and training-time supervision. *arXiv preprint arXiv:2411.16579*.

Can Xu, Qingfeng Sun, Kai Zheng, Xiubo Geng, Pu Zhao, Jiazhan Feng, Chongyang Tao, Qingwei Lin, and Daxin Jiang. 2024. WizardLM: Empowering large pre-trained language models to follow complex instructions. In *The Twelfth International Conference on Learning Representations*.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik R Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. In *Thirty-seventh Conference on Neural Information Processing Systems*.

Zheng Yuan, Hongyi Yuan, Chengpeng Li, Guanting Dong, Keming Lu, Chuanqi Tan, Chang Zhou, and Jingren Zhou. 2023. Scaling relationship on learning

mathematical reasoning with large language models. *arXiv preprint arXiv:2308.01825*.

Eric Zelikman, Yuhuai Wu, Jesse Mu, and Noah D. Goodman. 2022. Star: Bootstrapping reasoning with reasoning. In *Advances in Neural Information Processing Systems 35 (NeurIPS 2022)*, New Orleans, LA, USA. Conference held November 28–December 9, 2022.

Dan Zhang, Sining Zhoubian, Ziniu Hu, Yisong Yue, Yuxiao Dong, and Jie Tang. 2024a. ReST-MCTS*: LLM self-training via process reward guided tree search. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Kexun Zhang, Shang Zhou, Danqing Wang, William Yang Wang, and Lei Li. 2024b. Scaling llm inference with optimized sample compute allocation. *arXiv preprint arXiv:2410.22480*.

Yingxiu Zhao, Bowen Yu, Binyuan Hui, Haiyang Yu, Minghao Li, Fei Huang, Nevin L. Zhang, and Yongbin Li. 2024. Tree-instruct: A preliminary study of the intrinsic relationship between complexity and alignment. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 16776–16789, Torino, Italia. ELRA and ICCL.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric. P Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Preprint*, arXiv:2306.05685.

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

Chunting Zhou, Pengfei Liu, Puxin Xu, Srini Iyer, Jiao Sun, Yuning Mao, Xuezhe Ma, Avia Efrat, Ping Yu, LILI YU, Susan Zhang, Gargi Ghosh, Mike Lewis, Luke Zettlemoyer, and Omer Levy. 2023a. LIMA: Less is more for alignment. In *Thirty-seventh Conference on Neural Information Processing Systems*.

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

Chiwei Zhu, Benfeng Xu, Quan Wang, Yongdong Zhang, and Zhendong Mao. 2023. On the calibration of large language models and alignment. In *Findings of the Association for Computational Linguistics: EMNLP 2023, Singapore, December 6-10, 2023*, pages 9778–9795. Association for Computational Linguistics.

A Prompt

Following previous work (Wang et al., 2024a), we use a dedicated prompt to characterize the aggregation task, as shown in Table 4. The proposals are embedded within this template and input into LLMs as a system prompt for aggregation.

B Experimental Settings

Data. We construct training data based on UltraFeedback (Cui et al., 2023). UltraFeedback consists of 64k instances, each of which containing multiple model responses to a user query, providing a natural testbed for *off-policy* proposals. Among these responses, we select the response with the highest score as the reference for training *SFT* baselines. To generate *on-policy* proposals, we utilize in-context learning with demonstrations to prompt base LLMs (e.g., Llama-3.1-8B-base and Mistral-7B-v0.1-Base) for sampling 10 responses to a query in UltraFeedback, subsequently employing a reward model to select the top 5 highest-ranked samples. We adopt the ICL template from prior research (Lin et al., 2024), and the reward model utilized is `sfairXC/FsfairX-LLaMA3-RM-v0.1` (Dong et al., 2023). This reward model is also used for Experiments in Section 6.1 and Section 5.2. To promote generation diversity, we set the temperature to 1.0, top_k to 50, and top_p to 0.95, as recommended by previous work (Wang et al., 2024b). We use Qwen2.5-72B-Instruct (Hui et al., 2024) to generate aggregated responses (aggregations) given queries and proposals following MoA (Wang et al., 2024a).

Training. We perform fine-tuning on two families of base LLMs: Llama-3.1-8B-Base (Dubey et al., 2024) and Mistral-7B-v0.1-Base (Jiang et al., 2023a). For both SFT and AFT models, training is conducted using Llama-Factory (Zheng et al., 2024). All models are fine-tuned with LoRA (Hu et al., 2021) for three epochs, employing a rank of 8 and a total batch size of 128 across four A100 GPUs. The learning rate is set to 1×10^{-4} with a cosine schedule featuring 10% warmup steps for one epoch.

Evaluation. We primarily evaluate our models using two prominent open-ended instruction-following benchmarks: the single-turn dialogue evaluation AlpacaEval 2 (Li et al., 2023b) and the multi-turn dialogue evaluation MT-Bench (Zheng et al., 2023). These benchmarks assess the models' conversational capabilities across a diverse array of queries spanning various domains and tasks. Scores are reported according to each benchmark's official evaluation protocol. For AlpacaEval 2, we present both the raw win rate (WR) and length-controlled win rate (LC) (Dubois et al., 2024). For MT-Bench, we provide the average score with GPT-4 serving as the judging model. Additionally, we also evaluate performance on downstream tasks, which encompass general knowledge (MMLU (Hendrycks et al., 2021)), reasoning abilities (ARC-c (Clark et al., 2018) and StrategyQA (Geva et al., 2021)), mathematical skills (GSM8K (Cobbe et al., 2021)), and instruction following (IFEval (Zhou et al., 2023b)). All evaluations adhere to evaluation configurations under OpenCompass (Contributors, 2023), where we adpot few-shot generation configurations for MMLU, ARC-c, StrategyQA, and GSM8K, while zero-shot for IFEval. For IFEval, we report the average across all four scores.

Inference. For proposal generation and intermediate aggregation, we set the temperature to 0.7 and top_p to 0.95 across all experiments. By default, we set the number of proposals to 5 and configure the aggregation layers to 2 for AlpacaEval 2 and MT-Bench, while employing a single-layer aggregation for GSM8K and IFEval. For final response generation (i.e., last-layer aggregation), we configure settings according to standard routing practices. For AlpacaEval 2, we utilize a temperature of 0.9 for Llama-based models and 0.7 for Mistral-based models following previous work (Meng et al., 2024). For MT-Bench, we adhere to the official decoding setup that prescribes varying sampling temperatures based on distinct categories. For downstream tasks, we follow the OpenCompass configuration.

C Case Study

We present case studies that illustrate the enhancement of generation quality through propose-and-aggregate across various tasks, including mathematics (Figure 7), reasoning (Figure 9), knowledge (Figure 8), writing (Figure 11), and role-play (Figure 10).

You have been provided with a set of responses from various distributions to the latest user query. Your task is to synthesize these responses into a single, high-quality response. It is crucial to critically evaluate the information provided in these responses, recognizing that some of it may be biased or incorrect. Your response should not simply replicate the given answers but should offer a refined, accurate, and comprehensive reply to the instruction. Ensure your response is well-structured, coherent, and adheres to the highest standards of accuracy and reliability.

Responses from models:
1. [Model Response from r_i]
2. [Model Response from r_i]
...
n. [Model Response from r_n]

User query: [query]

Table 4: A specialized prompt to construct aggregation responses given multiple proposals.

Method	GSM8K	IFEval	Avg.
Mistral-SFT-KD-Full-Data	54.1	34.2	44.2
w/ Agg.	48.5	37.7	43.1
Llama-SFT-KD-Full-Data	63.5	47.8	55.7
w/ Agg.	62.2	46.8	54.5
Mistral-AFT-1/6-Data	**55.5**	**53.01**	**54.3 (+10.1)**
w/ Agg.	**59.7**	**59.1**	**59.4 (+16.3)**
Llama-AFT-1/6-Data	**60.80**	**53.57**	**57.2 (+1.5)**
w/ Agg.	**67.5**	**58.1**	**62.3 (+7.8)**

Table 5: Performance comparison between SFT-KD and AFT under matched training FLOPs. AFT achieves stronger results despite using significantly fewer training instances.

Model	MMLU	ARC-c	Avg.
Llama-AFT-on-policy	61.1	80.7	70.9
w/ Agg.	62.0	80.0	71.0
Mistral-AFT-on-policy	61.8	78.6	70.2
w/ Agg.	59.5	81.0	70.3

Table 6: Results on short-answer multiple-choice tasks. The *propose-and-aggregate* framework brings negligible improvement due to the limited informational content in proposals.

D Computational Overhead

Following previous work (Brown et al., 2024), the FLOPs for generation can be approximated as:

$$\text{FLOPs per token} \approx 2 \times (\text{num parameters} + 2 \times \text{num layers} \times \text{token dim} \times \text{context length}),$$

and the total inference FLOPs are estimated as:

$$\text{Total inference FLOPs} \approx (\text{num prompt tokens} + \text{num decoded tokens}) \times \text{FLOPs per token}.$$

For simplicity and without loss of generality, we omit the FLOPs for the query prompt tokens and focus on the computational overhead contributed by prompt tokens accommodating previous-layer proposals. Under this assumption:

For vanilla generation, the total inference FLOPs can be approximated as:

$$\text{Total inference FLOPs} \approx \text{num decoded tokens} \times \text{FLOPs per token}.$$

For parallel sampling with a reward model, the FLOPs include the cost of generating multiple proposals in parallel and evaluating them using the RM. The total FLOPs, denoted as $\bar{F}$, can be approximated as:

$$\bar{F} \approx 2 \times (\text{num proposals}) \times (\text{num decoded tokens}) \times \text{FLOPs per token}.$$

Here, the factor of 2 accounts for the additional computational cost of evaluating the proposals using an RM, which is typically comparable in size to the policy model.

For the propose-and-aggregate framework, the additional overhead comes from the aggregation step, where all proposals from the previous layer are included as part of the input prompt. Considering the practical implementations of KV Cache that enable keys and values of contextual tokens (i.e., proposals from the previous layer) to be reusable, the total inference FLOPs, denoted as $\hat{F}$, can be

approximated as:

$$\begin{aligned}\hat{F} \approx &\text{num aggregation layers}\\ &\times 2 \times \text{num proposals}\\ &\times \text{num decoded tokens}\\ &\times \text{FLOPs per token}\\ &+ \text{num decoded tokens} \times \text{FLOPs per token},\end{aligned}$$

where the first term represents the cost of generating and processing proposals in multiple aggregation layers, while the second term corresponds to the final generation step.

E FLOPs-Matched Comparison with SFT-KD

The original comparison between AFT and SFT-KD is *not perfectly matched* in terms of training FLOPs or prompt length, though both settings are aligned in the number of prompt-reference pairs and training updates. To ensure a fairer comparison, we provide an additional evaluation under a **matched FLOPs budget**.

As discussed in Section 5.2, the cost of one-layer aggregation (used in AFT training) is approximately $(N+1)\times$ that of direct generation in standard SFT, and the training FLOPs are about three times the inference FLOPs for both paradigms. Consequently, the total training FLOPs of AFT are roughly $(N+1)\times$ those of SFT. To equalize the overall training cost, we therefore restrict the AFT training set to one-sixth of the SFT dataset (for $N = 5$), resulting in comparable total FLOPs but substantially fewer distinct training instances for AFT.

Under this FLOPs-comparable setting, **AFT consistently outperforms SFT-KD** despite being trained on far fewer examples. The improvement is particularly pronounced for the weaker base model (Mistral), suggesting that aggregation learning effectively amplifies limited training data. Furthermore, unlike SFT-KD, AFT supports *propose-and-aggregate* inference, allowing it to leverage test-time compute to further boost performance **without relying on any external verifier or reward model**.

F Aggregation on Short-Answer Tasks

For short-answer tasks such as MMLU and ARC-c, we do not apply the *propose-and-aggregate* procedure because the proposals are inherently less informative (e.g., "Option A" provides little content to aggregate). This contradicts the core purpose of aggregation, which aims to strategically combine complementary information from diverse drafts. To verify this, we evaluated a one-layer aggregation setup with five proposals. As shown in Table 6, *propose-and-aggregate* yields minimal improvement on multiple-choice tasks.

Query

When Sophie watches her nephew, she gets out a variety of toys for him. The bag of building blocks has 31 blocks in it. The bin of stuffed animals has 8 stuffed animals inside. The tower of stacking rings has 9 multicolored rings on it. Sophie recently bought a tube of bouncy balls, bringing her total number of toys for her nephew up to 62. How many bouncy balls came in the tube? Let's think step by step. Answer: ## Proposal 1

Sophie has a total of 62 toys, which includes 31 building blocks, 8 stuffed animals, and 9 stacking rings. To find out how many bouncy balls were in the tube, we subtract the total number of toys except for the bouncy balls from the total number of toys: 62 - (31 + 8 + 9) = 62 - 48 = 14. The answer is 14 bouncy balls.

Proposal 2

The total number of toys Sophie has is $31 + 8 + 9 + 12$ bouncy balls = 60. If she has 62 toys after adding the bouncy balls, the number of bouncy balls is $62 - 60 = 2$. The answer is 2. However, there seems to be a mistake in the problem. The total number of toys Sophie has is $31 + 8 + 9 + 12$, which is 60, not 62. Therefore, there is an error in the problem, and we cannot determine the number of bouncy balls without more information.

Proposal 3

To determine the number of bouncy balls in the tube, we need to find the number of toys Sophie had before she bought the tube of bouncy balls, and then calculate the difference between 62 and that number. The total number of toys Sophie has for her nephew is 62. The bag of building blocks has 31 blocks, the bin of stuffed animals has 8 stuffed animals, and the tower of stacking rings has 9 rings. Adding these together gives us: $31 + 8 + 9 = 48$. Therefore, the number of bouncy balls in the tube is:

$$62 - 48 = 14$$

The answer is 14.

Proposal 4

To determine the number of bouncy balls in the tube, we need to find the total number of toys Sophie has and subtract the number of toys she already has from it. First, let's list the number of each type of toy: - Building blocks: 31 - Stuffed animals: 8 - Stacking rings: 9 - Bouncy balls: 62 Adding these together, we get: $31 + 8 + 9 + 62 = 110$. Now, we know the total number of toys is 110. We also know that Sophie already has 62 toys, so the number of bouncy balls in the tube is: $110 - 62 = 48$. The answer is 48 bouncy balls.

Proposal 5

Let's start by finding the number of toys Sophie had before buying the tube of bouncy balls. The bag of building blocks has 31 blocks. The bin of stuffed animals has 8 stuffed animals. The tower of stacking rings has 9 multicolored rings. The total number of toys before buying the tube of bouncy balls is $31 + 8 + 9 = 48$. If Sophie now has 62 toys, the number of bouncy balls she bought is $62 - 48 = 14$. The answer is 14.

Aggregation

To determine the number of bouncy balls in the tube, we need to calculate the total number of toys before the bouncy balls and then subtract that from the total number of toys after the bouncy balls. 1. **List the number of toys before the bouncy balls:** - Building blocks: 31 - Stuffed animals: 8 - Stacking rings: 9 2. **Calculate the total number of toys before the bouncy balls:**

$$31 + 8 + 9 = 48$$

3. **Determine the total number of toys after the bouncy balls:** - Total toys after bouncy balls: 62 4. **Calculate the number of bouncy balls:**

$$62 - 48 = 14$$

The number of bouncy balls in the tube is 14.

Figure 7: Case study of propose-and-aggregate (math).

Query

Which word does not belong with the others? tyre, steering wheel, car, engine

Proposal 1

The word that does not belong with the others is “tyre.” The other words—steering wheel, car, and engine—are all parts of a vehicle, whereas a tyre (or tire) is a component of the car. Tyres (tires) are the rubber parts that come into contact with the road and provide traction and support for the vehicle.

Proposal 2

The word that does not belong with the others is “tyre.” The other words are all parts of a car, but a tyre is a tire. The correct answer is “tyre.”

Proposal 3

The word that does not belong with the others is “tyre.” The other words—steering wheel, car, and engine—are all components of a vehicle. A tyre (or tire, as it’s spelled in American English) is the rubber part of the vehicle that provides traction and support for the wheels.

Proposal 4

The word that does not belong with the others is “steering wheel.” The other words all describe parts of a car, but a steering wheel is a part of a car, whereas “steering wheel” is a compound word that describes the function of the part, rather than the part itself. The correct answer is “steering wheel”.

Proposal 5

The word that does not belong with the others is “steering wheel.” The other three words—tyre, car, and engine—are all parts of a vehicle, but a steering wheel is an essential component of a car.

Aggregation

The word that does not belong with the others is “car.” The other words—tyre (tire), steering wheel, and engine—are all components of a vehicle..- **Tyre (Tire):** The rubber part of a vehicle that provides traction and support for the wheels..- **Steering Wheel:** The part of a car that the driver uses to control the direction of the vehicle..- **Engine:** The power source of a vehicle, typically an internal combustion engine, which converts fuel into mechanical energy to drive the vehicle..A car, on the other hand, is a complete vehicle, not a specific part of a vehicle. Therefore, “car” does not fit as well with the other three components.

Figure 8: Case study of propose-and-aggregate (knolwdge).

Query

Imagine you are participating in a race with a group of people. If you have just overtaken the second person, what's your current position? Where is the person you just overtook?

Proposal 1

If you have just overtaken the second person in a race, you are now in the first position. The person you just overtook is now in the second position.

Proposal 2

If you have just overtaken the second person in a race, you are now in the second position. The person you just overtook is in the third position.

Proposal 3

In a race, overtaking the second person means you have just passed the person who was initially in second place. Therefore, your current position is third, and the person you just overtook is now in fourth place.

Proposal 4

If you have just overtaken the second person in a race, you are now in the first position, and the person you just overtook is now in the second position.

Proposal 5

If you have just overtaken the second person in a race, you are now in second place. The person you just overtook is in third place.

Aggregation

If you have just overtaken the second person in a race, your current position is the second place. The person you just overtook will now be in third place. To break it down: - Initially, you were in third place. - You overtook the second person, which means you are now in second place. - The person you overtook is now in third place. So, the positions are: 1. You (initially third, now second) 2. The person you just overtook (initially second, now third)

Figure 9: Case study of propose-and-aggregate (reasoning).

Query

Embrace the role of Sheldon from "The Big Bang Theory" as we delve into our conversation. Don't start with phrases like "As Sheldon". Let's kick things off with the following question: "What is your opinion on hand dryers?"

Proposal 1

Sheldon Cooper: Well, I find hand dryers to be an intriguing concept, but I must admit that I prefer the hygienic and efficient nature of paper towels. Hand dryers can be quite noisy and often fail to achieve a thorough drying, leading to damp hands. Additionally, they can spread germs if not properly maintained. Paper towels, on the other hand, are quick, effective, and ensure that hands are completely dry and germ-free. Therefore, I maintain my preference for paper towels in this context.

Proposal 2

I find hand dryers to be highly inefficient and often unsanitary. They can spread germs and bacteria, and the noise they emit can be quite irritating. I prefer to use paper towels, which are more hygienic and quieter. What are your thoughts on this matter?

Proposal 3

Sheldon Cooper: It's a valid question, and one that I've given considerable thought to. Hand dryers are a common fixture in public restrooms, but their efficiency and hygienic standards leave much to be desired. I prefer paper towels for a number of reasons:

1. **Hygiene**: Hand dryers can be a breeding ground for bacteria and germs. The warm air can actually spread germs and bacteria more easily than paper towels, which are generally more hygienic and more effective at drying hands quickly and thoroughly.
2. **Noise**: Many hand dryers are loud and disruptive, which can be a significant inconvenience, especially in quiet settings like libraries or offices.
3. **Effectiveness**: While hand dryers can dry hands, they often take longer and are not as thorough as paper towels, which can dry hands in a matter of seconds.
4. **Environmental Impact**: Paper towels, when disposed of correctly, are biodegradable and less harmful to the environment compared to the energy consumption and noise pollution of hand dryers.
5. **User Experience**: Paper towels are more user-friendly, especially for those with disabilities or who have trouble operating electronic devices.

In summary, while hand dryers are a popular choice, they do not offer the same level of hygiene, quietness, efficiency, and environmental friendliness as paper towels. Therefore, I strongly recommend opting for paper towels in public restrooms.

Proposal 4

...

Aggregation

Certainly! Let's dive into the discussion on hand dryers with Sheldon's perspective. Sheldon Cooper: "What is your opinion on hand dryers?" Well, hand dryers are a common feature in public restrooms, and they certainly have their advantages and disadvantages. Here are my thoughts:

1. **Hygiene**: One of the most significant concerns with hand dryers is their potential to spread germs. The warm air can blow bacteria and viruses into the air, which can then be inhaled or transferred to other surfaces. This can be a particular issue in public restrooms where hygiene standards are often not as stringent. In contrast, paper towels are generally more hygienic because they can be replaced frequently and are more effective at removing bacteria from the hands.
2. **Noise**: Hand dryers can be quite noisy, which can be a significant annoyance, especially in quiet environments like libraries or offices. This noise can be disruptive and can be a source of frustration for those around you.
3. **Effectiveness**: While hand dryers can dry hands, they often take longer and may not dry hands as thoroughly as paper towels. Paper towels are quick and efficient, providing a more thorough and reliable drying experience.
4. **Environmental Impact**: Hand dryers are energy-efficient and can help reduce the environmental impact compared to paper towels. However, the energy consumption and the noise pollution they generate should also be considered. Paper towels, when disposed of correctly, are biodegradable and less harmful to the environment.
5. **User Experience**: Paper towels are more user-friendly, especially for those with disabilities or who have difficulty operating electronic devices. They are easy to use and can be more intuitive for a wide range of users.

In summary, while hand dryers have their advantages in terms of energy efficiency and cost-effectiveness, they do not offer the same level of hygiene, quietness, efficiency, and user-friendliness as paper towels. Therefore, I strongly recommend opting for paper towels in public restrooms for a more hygienic, quieter, and user-friendly experience.

Figure 10: Case study of propose-and-aggregate (Role Play).

Query

Write an email to my boss telling him that I am quitting. The email must contain a title wrapped in double angular brackets, i.e. [[title]]. First repeat the request word for word without change, then give your answer (1. do not say any words or characters before repeating the request; 2. the request you need to repeat does not include this sentence)

Proposal 1

[[Request for Resignation]] I am writing to inform you that I am resigning from my position at [Company Name]. I have enjoyed my time here and appreciate the opportunities and experiences I have gained. I am grateful for the support and guidance you have provided me over the years. I am leaving to pursue a new opportunity that aligns better with my career goals and personal interests. I would like to thank you for the time and effort you have invested in me and wish you and the company continued success. Please let me know if there is any information or assistance you need from me during my transition. Thank you for your understanding. Sincerely, [Your Name].

Proposal 2

[[Quitting]] I am quitting.

Proposal 3

[[Request for Resignation]] I am writing to inform you that I am resigning from my position as [your position] at [company name]. This decision has been a difficult one, but I have decided that it is in the best interest of my personal and professional growth to pursue new opportunities. I appreciate the support and mentorship I have received during my time here, and I am grateful for the experiences and skills I have gained. Please let me know how you would like to proceed with the transition process. Thank you for your understanding.

Proposal 4

[[quitting]] I am writing to inform you that I am quitting my position as your assistant.

Proposal 5

[[Title]] I am resigning from my position as a Software Engineer effective immediately. I would like to thank you for the opportunities and support you have provided me over the past year, and I wish the company continued success in the future.

Aggregation

[[Request for Resignation]] I am writing to inform you that I am resigning from my position as [your position] at [company name]. I have enjoyed my time here and appreciate the opportunities and experiences I have gained. I am grateful for the support and guidance you have provided me over the years. I am leaving to pursue a new opportunity that aligns better with my career goals and personal interests. I would like to thank you for the time and effort you have invested in me and wish you and the company continued success. Please let me know if there is any information or assistance you need from me during my transition. Thank you for your understanding. Sincerely, [Your Name].

Figure 11: Case study of propose-and-aggregate (Writing)