Staircase Streaming for Low-Latency Multi-Agent Inference

Junlin Wang [1 2] **Jue Wang** [2] **Zhen (Zach) Xu** [3] **Ben Athiwaratkun** [2]
Bhuwan Dhingra [1] **Ce Zhang** [2 3] **James Zou** [2 4]

Abstract

Recent advances in large language models (LLMs) opened up new directions for leveraging the collective expertise of multiple LLMs. These methods, such as Mixture-of-Agents, typically employ additional inference steps to generate intermediate outputs, which are then used to produce the final response. While multi-agent inference can enhance response quality, it can significantly increase the time to first token (TTFT), posing a challenge for latency-sensitive applications and hurting user experience. To address this issue, we propose *staircase streaming* for low-latency multi-agent inference. Instead of waiting for the complete intermediate outputs from previous steps, we begin generating the final response as soon as we receive partial outputs from these steps. Experimental results demonstrate that staircase streaming reduces TTFT by up to 93% while maintaining response quality.

1. Introduction

Large language models (LLMs) (Brown et al., 2020; OpenAI, 2023; Team et al., 2023; Dubey et al., 2024; Yang et al., 2025; DeepSeek-AI, 2024) have significantly advanced the field of natural language processing, showing remarkable capabilities across a wide range of tasks and domains. With the proliferation of LLMs and their impressive achievements, a new line of research has emerged: *multi-agent inference* (Du et al., 2023; Liang et al., 2023; Chen et al., 2023b; Wang et al., 2024b). It aims to harness the collective strengths of multiple LLMs to produce more reliable, consistent, and high-quality responses, matching or even surpassing state-of-the-art performance.

While multi-agent inference techniques can significantly enhance response quality by combining the outputs of multiple LLMs, they often incorporate additional inference steps. For example, Multi-Agent Debate (MAD) (Liang et al., 2023) uses several models to generate answers independently and then debate their answers; Mixture-of-Agents (MoA) (Wang et al., 2024b) employs proposers to generate initial responses and an aggregator to synthesize a higher-quality response from them. They both require to wait for all additional generations to finish before producing the final output, leading to an increased time to first token (TTFT). This latency is particularly problematic for real-time applications such as chatbots, where low latency is crucial for user experience. This naturally raises the question: *How can we leverage the strengths of multi-agent inference while minimizing such latency?*

To address this question, we propose *staircase streaming*, a novel approach for low-latency multi-agent inference. Instead of waiting for complete results from all the agents in the previous step, our approach begins streaming tokens once the first chunk of tokens becomes available from the preceding step. By leveraging partial results and overlapping the generation processes, staircase streaming effectively reduces latency while maintaining the benefits of multi-agent inference.

The intuition behind staircase streaming is based on two key observations: (1) *Effective partial prompting*: Jiang et al. 2023c demonstrate that even partial tokens within a prompt can effectively instruct LLM. Similarly, we found partial generation results can provide valuable, diverse viewpoints for other LLMs. (2) *Diminishing returns in subsequent outputs*: The most crucial information in a model's response is often front-loaded. The first portion of generation typically contains the main ideas, while subsequent parts generally add details, examples, or refinements to these core concepts.

Figure 1 illustrates an example of staircase streaming applied to MoA. In the normal case (a), the multi-agent sytem is bottlenecked by the proposer with the longest generation time, resulting in significant latency. With staircase streaming (b), the aggregator begins generating output as soon as a chunk of tokens is available from each proposer. Once the aggregator generates a chunk of tokens, it waits for the next chunk from the proposers, updates its prompt with these

[1]Duke University [2]Together AI [3]University of Chicago [4]Stanford University. Correspondence to: Junlin Wang <junlin.wang@duke.edu>.

Proceedings of the 43rd International Conference on Machine Learning, Seoul, South Korea. PMLR 306, 2026. Copyright 2026 by the author(s).

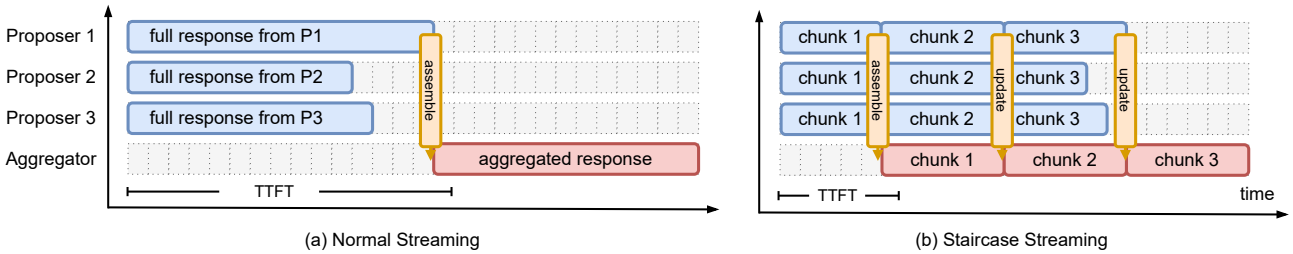

Figure 1. Comparison of normal streaming and staircase streaming using an MoA with 3 proposers and 1 aggregator. (a) In normal streaming, each LLM generates a full response before proceeding to the next step, leading to a longer TTFT. (b) Staircase streaming reduces TTFT by initiating the next step once the first chunks of proposed responses are available, enabling parallel processing between the proposers and the aggregator.

newly received tokens, and continues generating.

The evaluation results on the Arena-Hard and AlpacaEval benchmarks (Li et al., 2024a; Dubois et al., 2024) demonstrate that our staircase streaming can be effectively adapted to multi-agent inference methods, reducing latency up to 93% while maintaining the response quality.

2. Methodology

In this section, we first briefly overview token streaming. We then introduce staircase streaming, our approach that minimizes latency while leveraging multi-agent inference strengths. Additionally, we present a prefix caching optimized variant to further enhance efficiency.

2.1. Token Streaming

Token streaming allows autoregressive LMs to generate and return text incrementally, providing users with an early indication of content quality and enhancing the user experience. TTFT, measuring the latency between initiating a request and generating the first token, is a crucial metric for it.

However, multi-agent approaches often struggle with token streaming due to dependencies between multiple models. The first token becomes available after all preceding steps complete and the last step's prefill finishes. Depending on the generation length, this process can incur substantial latency, sometimes spanning dozens of seconds.

2.2. Staircase Streaming

Staircase streaming is proposed to enable low-latency multi-agent inference by streaming tokens between models as soon as partial results are available, rather than waiting for complete outputs. The key idea is to break the strict sequential dependencies between models, allowing a pipelined execution pattern where models process partial results immediately.

MoA is a typical multi-agent inference system where multiple proposer models generate initial responses based on a user prompt. These responses are then synthesized by an aggregator model to produce the final output. Algorithm 1 illustrates an adaptation of staircase streaming for MoA. The proposers and aggregator run in separate threads. Each proposer generates a chunk of tokens based on the user prompt and previously generated chunks, and then immediately sends the new chunk to the aggregator. The aggregator, after receiving chunks from proposers, updates its prompt by concatenating the new chunks. Then it generates the next chunk of the final output based on the updated prompt, which is streamed to the user. Meanwhile, the proposers process the next chunk of text in parallel. For multi-layer MoA systems, the same staircase streaming approach can be applied to each intermediate LLM, creating a nested staircase streaming pipeline.

The chunk sizes $C_{\mathrm{P},j}$ and $C_{\mathrm{A},j}$ control the granularity of the streaming process and can be adjusted to balance latency and computational efficiency. Smaller chunk sizes lead to more frequent updates but may require more prefill operations, while larger chunk sizes can increase TTFT. To achieve a

Algorithm 1 Staircase Streaming for MoA

Input: User prompt Q; Proposers $P_1, \ldots, P_N$; Aggregator A; Chunk sizes $C_{\mathrm{P},j}$ and $C_{\mathrm{A},j}$ ($j = 1, 2, \ldots$)
(1) THREAD PROPOSER ($i = 1, \ldots, N$):
  Initialize $j \leftarrow 0$
  while not end of response
    $j \leftarrow j + 1$
    $R_{i,j} \leftarrow P_i(Q + \cup_{k=1}^{j-1} R_{i,k})$ {Generate $C_{\mathrm{P},j}$ tokens}
    Send chunk $R_{i,j}$ to aggregator A
  end while
(2) THREAD AGGREGATOR:
  Initialize $j \leftarrow 0$
  while not end of response
    $j \leftarrow j + 1$
    Receive chunks $R_{1,j}, \ldots, R_{N,j}$ from all proposers or total number of proposers minus redundancy if redundancy is not 0.
    Update prompt $Q_{\mathrm{A},j}$ for the aggregator with proposed responses so far $[\cup_{k=1}^{j} R_{1,j}, \ldots, \cup_{k=1}^{j} R_{N,j}]$ with template in Table 5.
    $S_j \leftarrow A(Q_{\mathrm{A},j} + \cup_{k=1}^{j-1} S_k)$ {Generate $C_{\mathrm{A},j}$ tokens}
    (Streaming S_j to the user)
  end while
Output: $\cup_{k=1}^{j} S_k$

	ArenaHard Win Rate	AlpacaEval LC Win	TTFT second	TPS tokens/s
Gemma-2-9B-IT	40.6	48.5	$0.06_{\pm 0.01}$	$69.4_{\pm 0.4}$
MAD	50.8	55.8	$6.70_{\pm 2.3}$	$35.9_{\pm 4.4}$
+ staircase	45.9	55.2	$0.45_{\pm 0.0}$	$44.7_{\pm 6.6}$
MoA	47.5	56.8	$10.6_{\pm 3.4}$	$28.3_{\pm 8.9}$
+ staircase	48.3	55.1	$0.47_{\pm 0.1}$	$43.4_{\pm 6.5}$
+ staircase & prefix-cache	46.9	53.0	$0.45_{\pm 0.1}$	$45.0_{\pm 7.0}$

Table 1. Results of different inference methods.

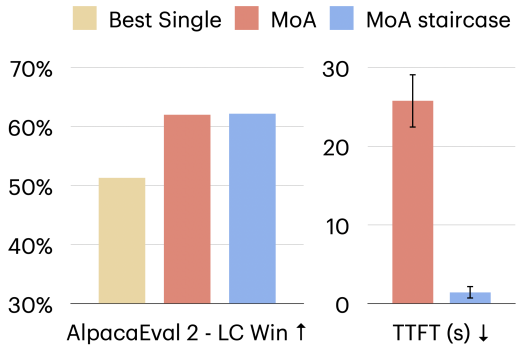

Figure 2. Results on Larger LLMs. The 'Best Single' model is WizardLM 8x22B. For MoA, the proposers include Qwen1.5-72B-Chat, Qwen1.5-110B-Chat, Wizard 8x22B, Mixtral-8x22B-Instruct-v0.1, and Llama-3-70B-Instruct. The aggregator is Qwen1.5-110B-Chat. TTFT was evaluated using the Together serverless endpoint, so the results may vary depending on server load.

better trade-off, we use smaller chunk sizes at the beginning and gradually increase them as the generation continues.

Following the similar principle, staircase streaming can be applied to general multi-agent systems by having each agent generate and stream chunks of tokens in parallel. These chunks are passed to the next agent or aggregator, which updates its prompt and continues the process.

Reducing TTFT In staircase streaming, TTFT is reduced since the aggregator starts generation with the first chunk $R_{i,1}$ from each proposer rather than waiting for the complete responses R_i. TTFT of MoA and staircase version can be represented as:

$$\text{TTFT}_{\text{normal}} = \max_{1 \le i \le N} \left(\sum_{j=1}^{\text{eos}} T_{R_{i,j}} \right) + T_{\text{prefill}} \tag{1}$$

$$\text{TTFT}_{\text{staircase}} = \max_{1 \le i \le N} \left(T_{R_{i,1}} \right) + T_{\text{prefill}} \tag{2}$$

where $T_{R_{i,j}}$ is the time taken by proposer i to generate the j-th chunk and T_{prefill} is the time taken by the aggregator to process the initial chunks.

2.3. Prefix-Caching Optimized Staircase Streaming

Prefix-caching enhances the efficiency of transformer models by reusing previously computed key-value pairs for identical prompt prefixes (Zheng et al., 2023). In staircase streaming, we can leverage this technique by appending new token chunks to the end of the prompt, keeping the prefix unchanged. This approach, whose prompt template is presented in Table 6, allows seamless updating of the staircase streaming prompt without requiring a full prefill computation. While this method may fragment responses from the same proposer, it can save more compute.

3. Evaluation

3.1. Setup

Benchmarks We evaluate on Arena-Hard (Li et al., 2024a) and AlpacaEval 2.0 (Dubois et al., 2024). For efficiency metrics, we assess the time to first token (TTFT) and the end-to-end tokens per second (TPS)[1] on a subset of Arena-Hard and AlpacaEval with 64 samples (a total of 128 samples).

Methods We incorporate staircase streaming to Mixture-of-Agents (MoA) and Multi-Agent Debate (MAD). We use Gemma-2-9B-IT (Team et al., 2024), LLaMA-3.1-8B-Instruct (Dubey et al., 2024), Mistral-7B-Instruct-v0.3 (Jiang et al., 2023a), and Qwen-1.5-7B-Chat (Bai et al., 2023); and use Gemma-2-9B-IT as the aggregator. We set the chunk size to 8 for the first chunk, increasing to 128 for the second chunk, and capping at 256 for proposers and 128 for the aggregator for later chunks. Benchmarks were conducted using the Together Inference API.[2] To minimize the effect of queuing times associated with the serverless API, we use redundancy of 2 – we skip the slowest 2 models – for the first chunk. To reduce the variance caused by network latency and server load, we run the aggregator model with vLLM on 1 H100 GPU for TTFT and TPS metrics. Detailed setup can be found in Section A.

3.2. Results

Inference Efficiency and Chat Benchmarks Table 1 presents the evaluation results. Our approach significantly reduces TTFT by up to 93% and increases TPS by up to 1.6x, enhancing the responsiveness while preserving the response quality of multi-agent inference. Staircase streaming is effective across MoA and MAD, showing its versatility. While the prefix-cache optimized variant slightly reduces the win rate, it further improves the efficiency. For tasks with critical

[1] TPS = The number tokens generated by the aggregator ÷ the time elapsed between sending the request and receiving the last token.

[2] https://api.together.ai/playground/chat

Model	MATH (%)	GSM8k (%)
Gemma-2-9B-IT	46.4	88.1
LLaMA-3.1-8B-Instruct	44.1	71.6
Mistral-7B-Instruct-v0.3	11.5	42.1
Qwen-1.5-7B-Chat	21.7	51.2
Qwen-2.5-7B-Instruct	70.9	90.4
MoA	47.0	78.1
+ staircase	46.6	82.4
MoA_version2	73.5	92.7
+ staircase	72.2	91.7

Table 2. Results of different inference methods on the MATH (Hendrycks et al., 2021) and GSM8k (Cobbe et al., 2021) datasets. *MoA_version2* replaces Qwen-1.5-7B-Chat with Qwen-2.5-7B-Instruct and uses Qwen-2.5-7B-Instruct as aggregator. While staircase results show slight degradation compared to MoA results on MATH, they improve over the best proposers for GSM8k.

information at the end (reasoning tasks), our method maintains performance due to larger proposer chunks allowing better aggregator capacity.

Reasoning Capability Table 2 shows the reasoning task results. MoA with staircase streaming improves over the best proposers, while maintaining an acceptable drop compared to the original MoA. Notably, on the stronger *MoA_version2* setup, staircase trails the full MoA by only 1.3 points on MATH and 1.0 point on GSM8k, indicating that even when the final answer typically appears at the end of a response, processing partial proposer outputs still preserves enough of the reasoning signal for the aggregator to reach the correct conclusion.

Scaling up Model Size Figure 2 shows AlpacaEval results for larger models. The 'Best Single' model achieved a 51% LC win rate. In contrast, MoA with 6 proposers achieved 62%, showing the benefits of leveraging the expertise from multiple LLMs. The staircase version achieved a similar LC win rate while maintaining reasonable TTFT, showing our method effectively scales to larger models, achieving good performance with lower latency.

Effect of Different Aggregators To verify that staircase streaming is not tied to a particular aggregator, we evaluate it with three additional aggregators: LLaMA-3.1-8B-Instruct, Mistral-7B-Instruct-v0.3, and Qwen-1.5-7B-Chat. Table 3 shows that staircase streaming tracks the full MoA closely across all aggregators, staying within roughly 1.5 points on both Arena-Hard and AlpacaEval2, and even slightly improving over full MoA on Arena-Hard with the Mistral aggregator. This suggests that the latency–quality trade-off offered by staircase streaming generalizes beyond our default Gemma-2-9B-IT aggregator and is not specific to any single backbone.

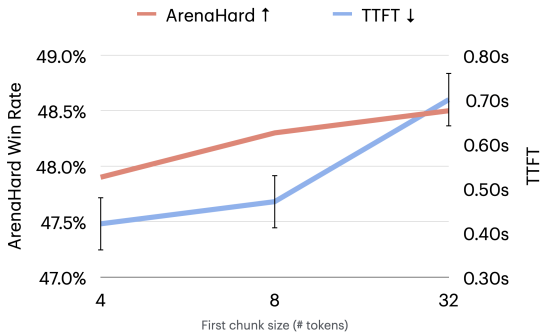

Figure 3. The impact of first chunk size on ArenaHard win rate and TTFT.

Effect of Chunk Sizes The first chunk size is a crucial hyperparameter for optimizing TTFT, as indicated in Equation (2). Figure 3 presents the impact of varying the first chunk size on response quality and TTFT. As the chunk size increases from 4 to 32 tokens, the win rate improves, while TTFT initially decreases but then increases for larger chunk sizes. This presents the trade-off between quality and latency, with a chunk size of 8 as the sweet point.

Additional Ablations on Chunk Sizes We further analyze the impact of varying chunk sizes individually. Using the configuration in Table 1 as a reference—specifically, *Start Chunk Size = 8*, *Second Chunk Size = 128*, and *Staircase Chunk Size = 256*—we vary each parameter, testing both lower and higher values. As shown in Table 4, performance remains fairly consistent across most configurations. However, setting the *Second Chunk Size* to 64 results in a notable performance degradation, suggesting that the early chunks carry important information that the aggregator relies on for effective synthesis.

Case Study Tables 8 and 9 present case studies demonstrating how staircase streaming effectively leverages the proposed chunks while maintaining coherence in context.

4. Related Work

Multi-Agent Inference A straightforward solution to leverage multiple LLMs is ranking outputs from different models. Jiang et al. (2023b) introduced PAIRRANKER, which performs pairwise comparisons on candidate outputs to select the best one. To alleviate the substantial computational costs of multi-LLM inference, other studies have explored training a *router* that predicts the best-performing model from a fixed set of LLMs for a given input (Wang et al., 2024a; Shnitzer et al., 2024; Yue et al., 2025; Zhang et al., 2025). Additionally, FrugalGPT (Chen et al., 2023c) proposed reducing the cost of using LLMs by employing different models in a cascading manner. To better lever-

Method	Arena-Hard	AlpacaEval2 (LC win rate)
LLaMA-3.1-8B-Instruct	28.0	26.06
Mistral-7B-Instruct-v0.3	16.3	19.88
Qwen-1.5-7B-Chat	12.6	16.58
MoA w/ LLaMA-3.1-8B-Instruct Aggregator	34.9	29.52
+ staircase	33.6	28.34
MoA w/ Mistral-7B-Instruct-v0.3 Aggregator	22.1	26.93
+ staircase	23.8	25.39
MoA w/ Qwen-1.5-7B-Chat Aggregator	20.7	25.54
+ staircase	19.1	24.40

Table 3. Performance of Arena-Hard and AlpacaEval2 (LC win rate). Results are shown for individual methods and combinations using the MoA framework with different aggregators.

Method	Start Chunk Size	Second Chunk Size	Staircase Chunk Size	Staircase Agg Chunk Size	Arena-Hard
First Chunk Ablat.	4	128	256	128	47.9
First Chunk Ablat.	32	128	256	128	48.5
Second Chunk Ablat.	8	64	128	128	40.7
Second Chunk Ablat.	8	256	512	128	42.6
Staircase Chunk Ablat.	8	128	128	128	48.7
Staircase Chunk Ablat.	8	128	512	128	48.7
MoA staircase (reference)	8	128	256	128	47.5

Table 4. Results of ablation studies and reference configuration for Arena-Hard. Each row shows different configurations of chunk sizes and the corresponding performance on Arena-Hard. We specifically study the effect of varying chunk sizes. Ablat. is abbreviation for ablation.

age the responses of multiple models, Jiang et al. (2023b) trained GENFUSER, a model designed to generate an improved response by capitalizing on the strengths of multiple candidates. Huang et al. (2024) proposed fusing the outputs of different models by averaging their output probability distributions. Another line of work focuses on collaborative inference, where multiple LLMs act as agents that collectively discuss and reason through given problems interactively. Du et al. (2023); Liang et al. (2023); Chan et al. (2023) established a mechanism for discussions among LLM-based agents. ReConcile (Chen et al., 2023b) adopts multi-agent discussion with weighted voting. Wang et al. (2024c) systematically compared collaborative approaches and found that a single agent with a strong prompt, including detailed demonstrations, can achieve comparable response quality to collaborative approaches. Shazeer et al. (2017) adopted a layered structure, using different LLMs as proposers to propose answers and another LLM to aggregate the final answer.

Efficient Inference Recent advancements in LLM inference optimization focus on enhancing system efficiency and resource utilization in order to reduce computation. vLLM (Kwon et al., 2023) addresses fragmentation with PagedAttention, which manages non-contiguous memory blocks efficiently. DeepSpeed-Inference (Aminabadi et al., 2022) combines GPU, CPU, and NVMe memory for high-throughput inference of diverse transformer models. FlexGen (Sheng et al., 2023) increases throughput with optimized GPU memory usage by integrating memory and computation from GPUs, CPUs, and disks, and quantizes weights to boost inference speed. Flash-Decoding (Dao et al., 2023) based on FlashAttention (Dao et al., 2022) accelerates long-context inference with parallel processing of KV pairs. On the algorithm side, KV-Cache optimization is a commonly studied topic. As the length of generated tokens increases, the KV cache expands significantly, resulting in high memory consumption and longer inference times, which recent work addresses via quantization and sparsification techniques (Hooper et al., 2024; Zhang et al., 2023; Xiao et al., 2023). Another inference speedup technique is speculative decoding (Leviathan et al., 2022; Chen et al., 2023a; Li et al., 2024b), which uses smaller models to generate multiple token guesses in parallel and then validates them, reducing computational load. Building on this idea, Staged Speculative (Spector & Re, 2023) organizes speculative outputs into a tree structure to enhance batch generation, and BiLD (Kim et al., 2023) proposes a fallback policy that allows the small model to defer to the target model when uncertain. To the best of our knowledge, there has been limited work on the inference efficiency of multi-agent systems.

5. Conclusion

In this paper, we introduced staircase streaming, an effective approach to reduce TTFT in multi-agent systems by streaming tokens incrementally between models. The key

innovation is to break the strict sequential dependencies inherent in existing multi-agent systems, enabling a pipelined execution pattern that leverages partial results to minimize idle waiting time. Our results showed that this approach not only maintains response quality but also significantly reduces TTFT by up to 93%.

6. Limitation

Our method requires modification to the original multi-agent algorithm. While these changes are generally minimal, they may impact performance. So necessary evaluation is needed before adapting staircase streaming to other multi-agent methods to ensure the quality does not change significantly.

Additionally, our work focuses primarily on TTFT as it is crucial for chat applications. However, we think chunk-by-chunk latency can be an interesting direction for future work.

References

Aminabadi, R. Y., Rajbhandari, S., Awan, A. A., Li, C., Li, D., Zheng, E., Ruwase, O., Smith, S., Zhang, M., Rasley, J., and He, Y. Deepspeed-inference: enabling efficient inference of transformer models at unprecedented scale. In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '22. IEEE Press, 2022.

Bai, J., Bai, S., Chu, Y., Cui, Z., Dang, K., Deng, X., Fan, Y., Ge, W., Han, Y., Huang, F., Hui, B., Ji, L., Li, M., Lin, J., Lin, R., Liu, D., Liu, G., Lu, C., Lu, K., Ma, J., Men, R., Ren, X., Ren, X., Tan, C., Tan, S., Tu, J., Wang, P., Wang, S., Wang, W., Wu, S., Xu, B., Xu, J., Yang, A., Yang, H., Yang, J., Yang, S., Yao, Y., Yu, B., Yuan, H., Yuan, Z., Zhang, J., Zhang, X., Zhang, Y., Zhang, Z., Zhou, C., Zhou, J., Zhou, X., and Zhu, T. Qwen technical report. *arXiv preprint arXiv: 2309.16609*, 2023.

Brown, T., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33: 1877–1901, 2020.

Chan, C.-M., Chen, W., Su, Y., Yu, J., Xue, W., Zhang, S., Fu, J., and Liu, Z. Chateval: Towards better llm-based evaluators through multi-agent debate. *arXiv preprint arXiv:2308.07201*, 2023.

Chen, C., Borgeaud, S., Irving, G., Lespiau, J.-B., Sifre, L., and Jumper, J. M. Accelerating large language model decoding with speculative sampling. *ArXiv*, abs/2302.01318, 2023a.

Chen, J. C.-Y., Saha, S., and Bansal, M. Reconcile: Round-table conference improves reasoning via consensus among diverse llms. *arXiv preprint arXiv:2309.13007*, 2023b.

Chen, L., Zaharia, M., and Zou, J. Frugalgpt: How to use large language models while reducing cost and improving performance. *arXiv preprint arXiv:2305.05176*, 2023c.

Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., Kaiser, L., Plappert, M., Tworek, J., Hilton, J., Nakano, R., et al. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

Dao, T., Fu, D. Y., Ermon, S., Rudra, A., and R'e, C. Flashattention: Fast and memory-efficient exact attention with io-awareness. *ArXiv*, abs/2205.14135, 2022.

Dao, T., Haziza, D., Massa, F., and Sizov, G. Flash-decoding for long-context inference, 2023.

DeepSeek-AI. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437*, 2024.

Du, Y., Li, S., Torralba, A., Tenenbaum, J. B., and Mordatch, I. Improving factuality and reasoning in language models through multiagent debate. *arXiv preprint arXiv:2305.14325*, 2023.

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

Dubois, Y., Galambosi, B., Liang, P., and Hashimoto, T. B. Length-controlled alpacaeval: A simple way to debias automatic evaluators. *arXiv preprint arXiv:2404.04475*, 2024.

Hendrycks, D., Burns, C., Kadavath, S., Arora, A., Basart, S., Tang, E., Song, D., and Steinhardt, J. Measuring mathematical problem solving with the math dataset. *NeurIPS*, 2021.

Hooper, C., Kim, S., Mohammadzadeh, H., Mahoney, M. W., Shao, Y. S., Keutzer, K., and Gholami, A. Kvquant: Towards 10 million context length llm inference with kv cache quantization. *ArXiv*, abs/2401.18079, 2024.

Huang, Y., Feng, X., Li, B., Xiang, Y., Wang, H., Qin, B., and Liu, T. Enabling ensemble learning for heterogeneous large language models with deep parallel collaboration. *arXiv preprint arXiv:2404.12715*, 2024.

Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., Chaplot, D. S., de las Casas, D., Bressand, F., Lengyel, G., Lample, G., Saulnier, L., Lavaud, L. R., Lachaux, M.-A., Stock, P., Scao, T. L., Lavril, T., Wang, T., Lacroix, T., and Sayed, W. E. Mistral 7b, 2023a.

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

Jiang, H., Wu, Q., Lin, C.-Y., Yang, Y., and Qiu, L. Llmlingua: Compressing prompts for accelerated inference of large language models. *arXiv preprint arXiv:2310.05736*, 2023c.

Kim, S., Mangalam, K., Moon, S., Malik, J., Mahoney, M. W., Gholami, A., and Keutzer, K. Speculative decoding with big little decoder. In *Neural Information Processing Systems*, 2023.

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

Leviathan, Y., Kalman, M., and Matias, Y. Fast inference from transformers via speculative decoding. In *International Conference on Machine Learning*, 2022.

Li, T., Chiang, W.-L., Frick, E., Dunlap, L., Wu, T., Zhu, B., Gonzalez, J. E., and Stoica, I. From crowdsourced data to high-quality benchmarks: Arena-hard and benchbuilder pipeline. *arXiv preprint arXiv:2406.11939*, 2024a.

Li, Y., Wei, F., Zhang, C., and Zhang, H. Eagle: Speculative sampling requires rethinking feature uncertainty. In *International Conference on Machine Learning*, 2024b.

Liang, T., He, Z., Jiao, W., Wang, X., Wang, Y., Wang, R., Yang, Y., Tu, Z., and Shi, S. Encouraging divergent thinking in large language models through multi-agent debate. *arXiv preprint arXiv:2305.19118*, 2023.

OpenAI. Gpt-4 technical report, 2023.

Shazeer, N., Mirhoseini, A., Maziarz, K., Davis, A., Le, Q., Hinton, G., and Dean, J. Outrageously large neural networks: The sparsely-gated mixture-of-experts layer. *arXiv preprint arXiv:1701.06538*, 2017.

Sheng, Y., Zheng, L., Yuan, B., Li, Z., Ryabinin, M., Chen, B., Liang, P., Ré, C., Stoica, I., and Zhang, C. Flexgen: high-throughput generative inference of large language models with a single gpu. In *Proceedings of the 40th International Conference on Machine Learning*, ICML'23. JMLR.org, 2023.

Shnitzer, T., Ou, A., Silva, M., Soule, K., Sun, Y., Solomon, J., Thompson, N., and Yurochkin, M. Large language model routing with benchmark datasets, 2024. URL https://openreview.net/forum?id=LyNsMNNLjY.

Spector, B. and Re, C. Accelerating llm inference with staged speculative decoding. *ArXiv*, abs/2308.04623, 2023.

Team, G., Anil, R., Borgeaud, S., Wu, Y., Alayrac, J.-B., Yu, J., Soricut, R., Schalkwyk, J., Dai, A. M., Hauth, A., et al. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805*, 2023.

Team, G., Riviere, M., Pathak, S., Sessa, P. G., Hardin, C., Bhupatiraju, S., Hussenot, L., Mesnard, T., Shahriari, B., Ramé, A., et al. Gemma 2: Improving open language models at a practical size. *arXiv preprint arXiv:2408.00118*, 2024.

Wang, H., Polo, F. M., Sun, Y., Kundu, S., Xing, E., and Yurochkin, M. Fusing models with complementary expertise. In *The Twelfth International Conference on Learning Representations*, 2024a. URL https://openreview.net/forum?id=PhMrGCMIRL.

Wang, J., Wang, J., Athiwaratkun, B., Zhang, C., and Zou, J. Mixture-of-agents enhances large language model capabilities. *arXiv preprint arXiv:2406.04692*, 2024b.

Wang, Q., Wang, Z., Su, Y., Tong, H., and Song, Y. Rethinking the bounds of llm reasoning: Are multi-agent discussions the key? *arXiv preprint arXiv:2402.18272*, 2024c.

Xiao, G., Tian, Y., Chen, B., Han, S., and Lewis, M. Efficient streaming language models with attention sinks. *ArXiv*, abs/2309.17453, 2023.

Yang, A., Li, A., Yang, B., Zhang, B., Hui, B., et al. Qwen3 technical report. *arXiv preprint arXiv:2505.09388*, 2025.

Yue, Y., Zhang, G., Liu, B., Wan, G., Wang, K., Cheng, D., and Qi, Y. Masrouter: Learning to route llms for multi-agent systems. *arXiv preprint arXiv:2502.11133*, 2025.

Zhang, H., Feng, T., and You, J. Router-r1: Teaching llms multi-round routing and aggregation via reinforcement learning. *arXiv preprint arXiv:2506.09033*, 2025.

Zhang, Z. A., Sheng, Y., Zhou, T., Chen, T., Zheng, L., Cai, R., Song, Z., Tian, Y., Ré, C., Barrett, C. W., Wang, Z., and Chen, B. H2o: Heavy-hitter oracle for efficient generative inference of large language models. *ArXiv*, abs/2306.14048, 2023.

Zheng, L., Yin, L., Xie, Z., Huang, J., Sun, C., Yu, C. H., Cao, S., Kozyrakis, C., Stoica, I., Gonzalez, J. E., et al. Efficiently programming large language models using sglang. *arXiv preprint arXiv:2312.07104*, 2023.

A. Detailed Setups

In the code implementation, there are five hyperparameters that can control the behaviors of staircase streaming: *first_chunk_size*, *second_chunk_size*, *chunk_size*, *aggregator_chunk_size*, and *redundancy*. When running staircase streaming, $C_{\mathrm{P},1}$ and $C_{\mathrm{A},1}$ would be set to *first_chunk_size*. $C_{\mathrm{P},2}$ and $C_{\mathrm{A},2}$ are set to *second_chunk_size*. Then for the rest of the chunks, $C_{\mathrm{P},j}$ and $C_{\mathrm{A},j}$ are set to *chunk_size* and *aggregator_chunk_size* respectively for $j > 2$.

Redundancy is introduced to control how many models' responses must be ready before the aggregator can start generating. For example, if there are five proposers and *Redundancy*=2, then that means for the first chunk, it only needs responses from three models before the aggregator can generate the first chunk. This granular setup allows practitioners to have maximum control over the trade-offs between TTFT and performance.

For our main results in Table 1, we use *first_chunk_size*=8, *second_chunk_size*=128, *chunk_size*=256, *aggregator_chunk_size*=128, and *redundancy*=2. These choices is carefully chosen from the ablation study, and generalize across different setups.

The improved TTFT and streaming experience comes at a cost of increased prompt token consumption. To mitigate this, we proposed a prefix-caching optimized version that can save a significant number of the tokens if the API services or the local inference engine supports it. Currently, more and more inference engines (Zheng et al., 2023; Kwon et al., 2023) and API services roll out the support prefix-caching.

B. Benchmarks Used

AlpacaEval 2 comprises 805 instructions that closely mirror real-world use cases. It implements length-controlled (LC) win rates to effectively neutralize length bias, a common confounding factor in language model evaluation. This metric has demonstrated remarkable alignment with human preferences, achieving a Spearman correlation of 0.98 with actual human evaluations.

Arena-Hard-Auto targets the evaluation of models on 500 challenging and demanding instructions submitted by real users in Chatbot Arena, thus maintaining a strong correlation with human preferences in complex scenarios. It contains difficult reasoning questions in the domain of math and coding.

MATH (Hendrycks et al., 2021) contains 5,000 challenging math problems that can showcase the model's reasoning capability.

C. Prompt Templates

We present the prompt templates used in our evaluation. For MoA experiments, we mainly followed the original paper by (Wang et al., 2024b). The staircase streaming templates are shown in Table 5 and Table 6. For MAD experiments, we adapted the templates to suit open-ended chat scenarios, as MAD is originally designed for tasks with short and deterministic answers, e.g. classification. The staircase streaming templates is shown in Table 7.

Table 5. Prompt template of staircase streaming for MoA.

Role	Content
system	You have been provided with a set of responses from various open-source models 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. $\{\cup_{k=1}^{j} R_{1,k}\}$ 2. $\{\cup_{k=1}^{j} R_{2,k}\}$ ... N. $\{\cup_{k=1}^{j} R_{N,k}\}$
user	{Q}

Table 6. Prompt template of prefix-caching optimized staircase streaming for MoA.

Role	Content
system	You have been provided with a set of responses from various open-source models to the latest user query in chunks. Your task is to synthesize these response chunks 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. As some responses may be incomplete yet, craft your synthesized response to allow for easy updating or expansion as new information becomes available. 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: Chunk 1: Model 1: $\{R_{1,1}\}$ ... Model N: $\{R_{N,1}\}$ Chunk 2: Model 1: $\{R_{1,2}\}$ ... Model N: $\{R_{N,2}\}$ ...
user	{Q}

Table 7. Prompt template of staircase streaming for MAD. MAD puts the model's own output before the debate prompt, here we assueme it's the 'N'-th model.

Role	Content
user	{Q}
assistant	$\{\{\cup_{k=1}^{j} R_{N,k}\}\}$
user	These are the responses to the query from other agents: One agent solution: $\{\cup_{k=1}^{j} R_{1,k}\}$ One agent solution: $\{\cup_{k=1}^{j} R_{2,k}\}$ ... One agent solution: $\{\cup_{k=1}^{j} R_{N-1,k}\}$ Using the responses from other agents as additional information, can you provide your response to the query? The original query is {Q}

Table 8. Case study: staircase streaming of MoA for reasoning task. We pick an example from the MATH dataset. Different colors indicate different chunks. We omit some of the generations to fit into one page. Notice that even for reasoning tasks where important information (answer) is usually located at the end, our staircase setup can still synthesize the correct answer.

Role	Content
User	Find the smallest integer that satisfies the inequality: [\n(x-5)(x+5)<0.]
Gemma-2-9B-IT	Here's how to solve the inequality:**1. Understand the Sign of the Product**The inequality $(x-5)(x+5)<0$ tells us that the product of the two factors is negative. This happens when one factor is positive and the other is negative.**2. Consider the Cases*** **Case 1:** $x-5>0$ and $x+5<0$. This leads to $x>5$ and $x<-5$, which has no solutions since a number cannot be both greater than 5 and less than -5.* **Case 2:** $x-5<0$ and $x+5>0$. This leads to $x<5$ and $x>-5$.**3. Combine the Solutions**The solution to the inequality is $-5<x<5$.**4. Find the Smallest Integer**The smallest integer that satisfies this inequality is -4.**Answer:** $\boxed{-4}$
LLaMA-3.1-8B-Instruct	To solve the inequality ((x-5)(x+5)<0), we need to find the values of (x) for which the expression is less than zero.First, let's find the critical points by setting each factor equal to zero and solving for (x):1. $(x-5=0 \implies x=5)$2. $(x+5=0 \implies x=-5)$These are the critical points where the expression changes sign. Now, we need to test the intervals between and beyond these points to see where the expression is negative.The intervals to test are:- (x < -5)- (-5 < x < 5)- (x > 5)Let's test a value from each interval:- For (x < -5), let's test (x = -10): ((-10 - 5)(-10 + 5) = (-15)(-5) = 75 > 0)- For (-5 < x < 5), let's test (x = 0): ((0 - 5)(0 + 5) = (-5)(5) = -25 < 0)- For (x > 5), let's test (x = 10): ((10 - 5)(10 + 5) = (5)(15) = 75 > 0)From the tests, we see that the expression is negative only in the interval (-5 < x < 5).Therefore, the solution to the inequality is (-5 < x < 5), which can be written as (-5 < x < 5).
Mistral-7B-Instruct-v0.3	The smallest integer that satisfies the inequality $(x-5)(x+5)<0$ is $x=4$.Here's the reasoning:1. The product of two numbers is less than zero if and only if one number is positive and the other is negative.2. The expression $(x-5)(x+5)$ will be positive when $x>5$ or $x<-5$ (since both factors are of the same sign), and negative when $-5<x<5$ (since one factor is positive and the other is negative).3. To find the smallest integer that satisfies the inequality, we look for the smallest integer within the interval $-5<x<5$.4. Testing the integers in this interval, we find that $x=4$ is the smallest integer that makes the inequality true, since $(4-5)(4+5)=-9$ is less than zero.So, the final answer is $\boxed{4}$.
Qwen-1.5-7B-Chat	...
Aggregated (Gemma-2-9B-IT)	Here's how to find the smallest integer solution to the inequality $(x-5)(x+5)<0$:**1. Understand the Sign of the Product**The inequality tells us that the product of the two factors is negative. This happens when one factor is positive and the other is negative.**2. Consider the Cases*** **Case 1:** $x-5>0$ and $x+5<0$. This leads to $x>5$ and $x<-5$, which has no solutions since a number cannot be both greater than 5 and less than -5.* **Case 2:** $x-5<0$ and $x+5>0$. This leads to $x<5$ and $x>-5$.**3. Combine the Solutions**The solution to the inequality is $-5<x<5$.**4. Find the Smallest Integer**The smallest integer that satisfies this inequality is -4.**Answer:** $\boxed{-4}$

Table 9. Case study: staircase streaming of MoA for reasoning task. We pick an example from the MATH dataset. Different colors indicate different chunks. We omit some of the generations to fit into one page. Notice that even for reasoning tasks where important information (answer) is usually located at the end, our staircase setup can still synthesize the correct answer. This example contains more chunks.

Role	Preference	Content
User	-	For how many different values of x does $x^5 = x^4 + 72x^3$?
Gemma-2-9B-IT	0.00	Here's how to solve the problem:**1. Rearrange the equation:**Subtract $x^4 + 72x^3$ from both sides to get a standard polynomial equation:$x^5 - x^4 - 72x^3 = 0$**2. Factor out a common factor:**Notice that all the terms have a common factor of x^3:$x^3(x^2 - x - 72) = 0$**3. Solve for x:**This equation has two parts:* **Part 1:** $x^3 = 0$. This has one solution: $x = 0$.* **Part 2:** $x^2 - x - 72 = 0$. We can solve this quadratic equation using the quadratic formula: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ Where $a = 1$, $b = -1$, and $c = -72$. Plugging in these values, we get: $x = \frac{1 \pm \sqrt{(-1)^2 - 4 \cdot 1 \cdot -72}}{2 \cdot 1}$ $x = \frac{1 \pm \sqrt{289}}{2}$ $x = \frac{1 \pm 17}{2}$ This gives us two more solutions: $x = 9$ and $x = -8$.**4. Combine the solutions:**The total number of different solutions is 3: $x = 0$, $x = 9$, and $x = -8$.**Answer:** $\boxed{3}$
LLaMA-3.1-8B-Instruct	0.00	To find the values of x that satisfy the equation $x^5 = x^4 + 72x^3$, we can rearrange the equation to get $x^5 - x^4 - 72x^3 = 0$. This is a polynomial equation of degree 5, and it can be factored by taking out a common factor of x^3 from all the terms, giving us $x^3(x^2 - x - 72) = 0$.Now, we have two factors: x^3 and $(x^2 - x - 72)$. The first factor, x^3, equals zero when $x = 0$. For the second factor, $(x^2 - x - 72)$, we can factor it ...is $9^5 = 59049$, but the right-hand side is$9^4 + 72(9)^3 = 6561 + 46656 = 52417$. This means that $x = 9$ is not a ...The number of different values of x that satisfy the equation is 2. $\boxed{2}$
Mistral-7B-Instruct-v0.3	0.16	To find the values of x that satisfy the equation $x^5 = x^4 + 72x^3$, we can rearrange the equation to form a quadratic equation in x^3.First, subtract x^4 from both sides:$x^5 - x^4 = 72x^3$Next, factor out x^3 from the left side:$x^3(x^2 - x) = 72x^3$Now, divide both sides by x^3:$x^2 - x = 72$Finally, multiply both sides by x to get a quadratic equation in x:$x^3 - x^2 = 72x$This is a cubic equation in x^3, ...we can express x as a function of x:$x = \frac{3}{2} + \sqrt{72x + 81/4}$Square both sides to eliminate the square root:$x^2 = \frac{9}{4} + 72x + \frac{81}{16} + 2\sqrt{72x + 81/4} \cdot \sqrt{72x + 81/4}$Simplify the right side:$x^2 = \frac{9}{4} + 72x + \frac{81}{16} + 144\sqrt{(72x + 81/4)}$ ...This is a quartic equation in x^2, which is difficult to solve analytically. However, ...Now, square both sides again to eliminate the square root: ...Unfortunately, it is not possible to find a single answer formula for the solutions of this equation. Instead, we can use numerical methods or a computer algebra system to find the real solutions. ...
Qwen-1.5-7B-Chat	0.03	...
Aggregated (Gemma-2-9B-IT)	0.33	Here's how to solve the problem:**1. Rearrange the equation:**Subtract x^4 and $72x^3$ from both sides to get a standard polynomial equation:$x^5 - x^4 - 72x^3 = 0$**2. Factor out a common factor:**Notice that all the terms have a common factor of x^3:$x^3(x^2 - x - 72) = 0$**3. Solve for x:**This equation has two parts:* **Part 1:** x^3 = 0. This has one solution: $x = 0$.* **Part 2:** $x^2 - x - 72 = 0$. We can solve this quadratic equation using the quadratic formula: $x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$ Where $a = 1$, $b = -1$, and $c = -72$. Plugging in these values, we get: $x = \frac{1 \pm \sqrt{(-1)^2 - 4 \cdot 1 \cdot -72}}{2 \cdot 1}$ $x = \frac{1 \pm \sqrt{289}}{2}$ $x = \frac{1 \pm 17}{2}$ This gives us two more solutions: $x = 9$ and $x = -8$.**4. Combine the solutions:**The total number of different solutions is 3: $x = 0$, $x = 9$, and $x = -8$.**Answer:** $\boxed{3}$