

Distribution-Aware and Uncertainty-Aware Scheduling for LLM Inference

Anonymous Authors¹

Abstract

In Large Language Model (LLM) inference, traditional First-Come-First-Served (FCFS) scheduling leads to Head-Of-Line (HOL) blocking and degraded quality of service. Existing work often applies Shortest Job First (SJF) principles to reduce average latency, which requires the scheduler to accurately predict the output length of inference requests. However, the stochastic decoding strategies in LLMs introduce output variability, making accurate prediction challenging. In this paper, we revisit this bottleneck and propose that the output length of LLMs can be fitted using specific probability distributions. By predicting the entire output length distribution rather than a single point estimate, our approach systematically captures the inherent dynamism in LLM outputs. To address both the intrinsic uncertainty of the distribution itself and the estimation errors introduced by the predictor, we further propose a risk-adaptive scheduling strategy. We apply our DUA scheduler to popular LLMs and conduct comprehensive evaluations across multiple datasets in both online and offline scenarios, achieving significant improvements: $4.73\times$ reduction in per-token latency for online inference and $2.34\times$ speedup in offline data generation. We believe this paradigm can be applied to other scenarios involving LLM output length prediction.

1. Introduction

Large Language Models (LLMs) are rapidly emerging as the cornerstone of modern artificial intelligence applications, with use cases spanning chatbots, content generation, scientific reasoning, and beyond. Popular LLM services such as ChatGPT (OpenAI, 2026), Gemini (Google, 2026) and Claude (Anthropic, 2026) process billions of inference

¹Anonymous Institution, Anonymous City, Anonymous Region, Anonymous Country. Correspondence to: Anonymous Author <anon.email@domain.com>.

Preliminary work. Under review by the International Conference on Machine Learning (ICML). Do not distribute.

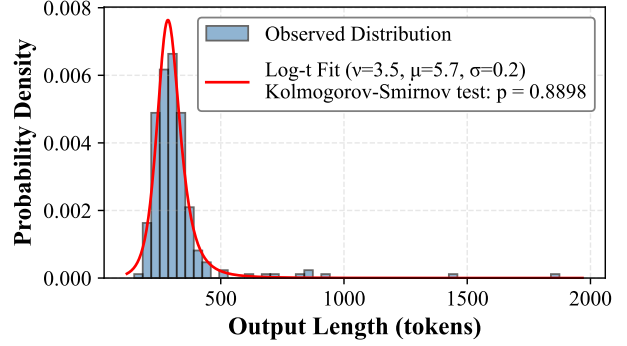


Figure 1. Output length distribution for the *first prompt* in LMSYS-Chat-1M dataset (Zheng et al., 2023a) (256 generations), while the red curve shows the fitted log-t location-scale distribution.

requests daily (Chatterji et al., 2025). To efficiently handle these inference workloads while maintaining Quality of Service (QoS), efficient scheduling strategies are essential.

Challenges. LLM serving systems typically employ First-Come-First-Served (FCFS) scheduling, which suffers from severe Head-of-Line (HOL) blocking (Fu et al., 2024). In such systems, long requests block shorter ones in the queue, resulting in increased latency and degraded system throughput. Recent efforts have explored Shortest-Job-First (SJF)-like policies to mitigate this issue by prioritizing requests with shorter predicted outputs (Qiu et al., 2024; Jin et al., 2023; Fu et al., 2024). However, these approaches face a fundamental challenge: the output length depends on *both* the input content and *stochastic* decoding strategies, making accurate length prediction inherently difficult or infeasible (Kwon et al., 2023a; Yu et al., 2022). Furthermore, misclassifying a long request as short can be particularly detrimental, as it reintroduces the very HOL blocking problem that SJF strategies aim to eliminate.

Recent work attempts to address this uncertainty through various strategies. Some methods predict ranking scores rather than exact lengths (Fu et al., 2024), while others employ frequent iterative predictions during generation, combined with request preemption (Shahout et al., 2025). However, these approaches either fail to explicitly model and effectively handle the underlying uncertainty, or incur sub-

stantial computational and memory overhead. This raises a critical question: *How can we design scheduling policies that effectively handle uncertainty while maintaining computational efficiency for LLM inference?*

Our Solution. Our key insight is that, while prompt output lengths exhibit uncertainty, they collectively follow identifiable statistical patterns. Therefore, we model the output length *distributions* and leverage the *uncertainty information* for scheduling, rather than attempting to avoid it. As illustrated in Figure 1, output lengths for the same prompt follow a long-tailed distribution that can be effectively modeled using a log-t (Cassidy et al., 2010) distribution. We validate this fit using the Kolmogorov-Smirnov (KS) test (Massey Jr, 1951), which yields a p-value of 0.8898, indicating that the observed data is consistent with the fitted distribution.

We employ a lightweight model (i.e., a fine-tuned DeBERTa-v3-base (He et al., 2021)) to extract prompt semantics and feed them into an MLP to predict distribution parameters. We derive scheduling scores from these predicted distributions. Notably, we account for prediction error and potential risks (i.e., the risk that prompts predicted to generate short outputs actually produce long ones) when computing scores, and employ an adaptive algorithm to manage this risk. We propose DUA a Distribution-Aware and Uncertainty-Aware scheduling strategy.

We comprehensively evaluate DUA across both online and offline scenarios, using multiple datasets and tasks, including chatbot services and data generation. Table 1 highlights our main results: DUA achieves a $4.73\times$ reduction in average per-token latency for online inference and a $2.34\times$ speedup in offline data generation.

Contributions. We make the following contributions:

- We identify inherent output uncertainty as the fundamental limitation of existing scheduling methods.
- To our knowledge, we are the first to employ distribution estimation rather than point estimation for LLM output length prediction, and evaluate multiple distribution families.
- We present DUA, which quantifies risk and adaptively adjusts scheduling priorities (detailed in Sec.3).
- We design asynchronous and batched prediction mechanisms to enable efficient scheduling.
- We conduct extensive experiments including ablation studies and open-source our code and models¹.

¹<https://github.com/>

Table 1. Performance comparison on online (Chatbot) and offline (Synthetic Data Generation, SDG) workloads. SDG metrics: time to generate 3k samples and samples generated within 3 minutes.

Tasks	Metrics	FCFS	DUA(Ours)
Chatbot	Per-token Latency (s)	3.17	0.67
	Time-to-first-token (s)	112.39	41.81
SDG	Time for 3k samples (s)	229.37	98.12
	#Samples within 3 min	2292	4762

2. Related Work

LLM Serving Systems. Modern systems such as vLLM (Kwon et al., 2023b), TensorRT-LLM (NVIDIA, 2023), and Text Generation Inference (TGI) (Hugging Face, 2023) have gained widespread adoption. However, they primarily rely on First-Come-First-Served (FCFS) scheduling, which leads to significant Head-of-Line (HOL) blocking under heavy workloads (Qiu et al., 2024; Fu et al., 2024; Shahout et al., 2025). Continuous Batching (Yu et al., 2022) is a critical advancement that prevents long-running requests from blocking entire batches. However, queue-level blocking remains an open challenge: long-running requests can still block shorter ones waiting in the queue, leading to increased average latency and reduced system throughput.

Scheduling for LLM Inference. To mitigate queue-level blocking, recent studies have applied Shortest-Job-First (SJF)-like policies, which are considered promising. Speculative Shortest-Job-First (SSJF) (Qiu et al., 2024) trains a lightweight proxy model to predict request lengths. Several other studies (Jin et al., 2023; Zheng et al., 2023b) have explored similar strategies based on output length prediction. However, as discussed earlier, the inherent uncertainty in LLM outputs makes accurate length prediction difficult.

To mitigate this uncertainty, Learning-to-Rank (LTR) (Fu et al., 2024) introduces Kendall’s Tau as a ranking metric and predicts ranking scores rather than exact lengths. Nevertheless, this approach still relies on point estimates and does not fully capture the variability in output lengths. A potential optimization pathway is preemptive scheduling. TRAIL (Shahout et al., 2025) employs an MLP to predict output length after each token generation, and performs preemption accordingly to mitigate potential losses. However, this preemptive approach incurs substantial overhead due to frequent iterative predictions and additional preemption costs (e.g., KV cache management).

LLM Output Length Prediction. Predicting LLM output length has applications beyond scheduling, such as cache management (Apple, 2024; Gim et al., 2024) and cost estimation (Piotrowski et al., 2025). Existing methods typically predict lengths from input embeddings or refine predictions

iteratively during generation. The core limitation is that point estimates inherently clash with the stochastic nature of autoregressive decoding.

3. Method

Our scheduling approach consists of three key components: (1) selecting and fitting appropriate distributions to model the LLM output length, (2) training a predictor to estimate the distribution parameters of output length for each inference request, and (3) designing an adaptive scheduling algorithm based on the predicted distributions.

3.1. Distribution Fitting for Output Length Modeling

The inference process of large language models (LLMs) is inherently stochastic. At each decoding step, the model samples from a probability distribution over the vocabulary:

$$P_\tau(x_t = v \mid x_{<t}) = \frac{\exp(z_v/\tau)}{\sum_{v' \in V} \exp(z_{v'}/\tau)} \quad (1)$$

where z_v is the logit for token v , and τ is the temperature parameter. Due to this stochastic decoding process, the same input prompt may yield different outputs.

Distribution Selection. To investigate the distribution of output lengths for identical prompts, we sample 1K prompts from the LMSYS-Chat-1M dataset and generate 100 responses for each. Across these 1K distributions, the average skewness is 3.10, and the mean Coefficient of Variation (CV) is 1.09, with 78.6% of them > 1 . The top 10% of output lengths account for 35.7% of the total length, while the average P90/P50 and P99/P50 ratios reach 4.62 and 10.77, respectively. These metrics collectively demonstrate that the output length distribution for a given prompt exhibits pronounced *heavy-tailed* characteristics (Heckert et al., 2002).

To identify suitable distribution families, we evaluate several common heavy-tailed distributions on the empirical distributions obtained above. We assess the goodness of fit using the Kolmogorov-Smirnov (KS) test, where $p > 0.05$ is conventionally considered to indicate adequate fit (Fisher, 1925; D’Agostino & Stephens, 1986; Alasmar et al., 2021).

We focus on the log-t distribution, defined as follows: if $Y \sim t(\nu)$ is a standard Student’s t-distribution (Student, 1908) with ν degrees of freedom, then $X = \exp(\mu + \sigma Y)$ follows a log-t distribution, denoted $X \sim \text{Log-t}(\mu, \sigma, \nu)$. Recall that the Probability Density Function (PDF) of the standard t-distribution is:

$$t_\nu(y) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi} \Gamma(\frac{\nu}{2})} \left(1 + \frac{y^2}{\nu}\right)^{-\frac{\nu+1}{2}} \quad (2)$$

Table 2. Goodness of fit for common long-tailed distributions (1,000 prompts, 100 generations each). KS pass rate indicates the percentage of prompts where the fitted distribution passes the KS test ($p > 0.05$).

Distribution	# Parameters	KS Pass Rate
Log-t	3	93.1%
Log-t ($\nu = 3.5$)	2	90.6%
Log-normal	2	60.3%
Gamma	2	32.0%
Weibull	2	19.0%
Exponential	1	10.7%

The PDF of the log-t distribution is then given by:

$$f(x \mid \mu, \sigma, \nu) = \frac{1}{\sigma x} \cdot t_\nu\left(\frac{\ln x - \mu}{\sigma}\right) \quad (3)$$

We fit distributions via Maximum Likelihood Estimation (MLE). Given observed data $\{x_1, x_2, \dots, x_n\}$, the log-likelihood function is:

$$\ell(\mu, \sigma, \nu) = \sum_{i=1}^n \ln f(x_i \mid \mu, \sigma, \nu) \quad (4)$$

Substituting the density function yields:

$$\ell(\mu, \sigma, \nu) = \sum_{i=1}^n \left[\ln t_\nu\left(\frac{\ln x_i - \mu}{\sigma}\right) - \ln \sigma - \ln x_i \right] \quad (5)$$

We employ the L-BFGS-B optimizer to estimate all three parameters simultaneously:

$$\hat{\mu}, \hat{\sigma}, \hat{\nu} = \arg \max_{\mu \in \mathbb{R}, \sigma > 0, \nu > 0} \ell(\mu, \sigma, \nu) \quad (6)$$

Additionally, we evaluate a two-parameter variant with fixed ν . Through empirical evaluation, we find that $\nu = 3.5$ yields the best fitting performance:

$$\hat{\mu}, \hat{\sigma} = \arg \max_{\mu \in \mathbb{R}, \sigma > 0} \ell(\mu, \sigma, \nu = 3.5) \quad (7)$$

Table 2 summarizes the parameter count and fitting performance for several common heavy-tailed distributions. The three-parameter log-t achieves the highest KS pass rate (93.1%), followed by the two-parameter log-t with fixed $\nu = 3.5$ (90.6%). In contrast, other distributions exhibit substantially poorer fit quality. While distributions with more parameters can provide more flexible fits, they also increase the complexity of the prediction model, incurring higher overhead during both training and inference. Balancing efficiency and accuracy, we adopt the two-parameter log-t distribution with $\nu = 3.5$ for subsequent experiments. We present ablation studies comparing end-to-end performance across these distributions in Section 5.4.

3.2. Distribution Parameter Prediction Model

With the distribution determined, we develop a predictive model to estimate distribution parameters (μ, σ) for each incoming prompt, thereby providing the necessary information for subsequent distribution-aware scheduling.

Model Architecture. The architecture of our predictor consists of three components:

(1) Encoder: We employ DeBERTa-v3-base (86M backbone) (He et al., 2021; 2023) to encode input prompts into contextualized embeddings. We select this encoder for its high performance on semantic understanding tasks and computational efficiency compared to larger models.

(2) Feature Extraction: To capture multi-level semantic information, we employ a *multi-pooling* fusion strategy that combines the *CLS* token, *mean-pooled*, and *max-pooled* representations. This combination allows the model to leverage complementary information: the CLS token encodes global semantics, mean pooling captures average contextual information, and max pooling highlights salient features.

(3) Parameter Prediction: We design separate prediction heads for μ and σ . Each head consists of an MLP with three hidden layers (256, 256, 128) and dropout for regularization. Since the empirical distribution of σ is right-skewed, we predict in a transformed space: $\tilde{\sigma} = \log(1 + \sigma)$.

Training Strategy. We perform z-score normalization on both μ and $\tilde{\sigma}$ to accelerate convergence. The model is trained using Mean Squared Error (MSE) loss:

$$\mathcal{L} = \frac{1}{N} \sum_{i=1}^N \left[(\hat{\mu}_i - \mu_i)^2 + (\hat{\sigma}_i - \tilde{\sigma}_i)^2 \right] \quad (8)$$

where $\hat{\mu}_i$ and $\hat{\sigma}_i$ denote the predicted values for prompt i , and $\mu_i, \tilde{\sigma}_i$ are ground-truth targets.

We employ a two-stage training strategy. In Stage 1 (epochs 1-6), *all* parameters are optimized with a learning rate of $2e-5$. In Stage 2 (epochs 7-9), we *freeze* the encoder and fine-tune only the prediction heads with a learning rate of $5e-5$. This strategy preserves the generalization capability of the pre-trained encoder while allowing refined tuning of the prediction heads. We provide full implementation details in our open-source code².

On the test set, the trained model achieves R^2 values of 0.82 and 0.76 for μ and σ , respectively.

3.3. Scheduling Algorithm

With the distribution prediction framework established, we now describe how to leverage the predicted distribution

parameters for scheduling.

For the stochastic scheduling model, given the execution time distributions of all jobs, Shortest Expected Processing Time (SEPT) is a classical strategy (Weber, 1983). However, applying this approach faces the following challenges:

1. While the log-t distribution provides a reasonable fit, it does not perfectly capture the true output length distribution.
2. Distribution parameters are estimated from the predictor and inevitably contain errors.
3. Requests predicted to be long may suffer from starvation.

The first two issues highlight an unavoidable discrepancy between the predicted and actual output length distributions. In particular, when a genuinely long request is mispredicted as short, it blocks subsequent requests in the queue. This problem is further exacerbated by the fact that LLM outputs naturally exhibit heavy-tailed behavior. These observations motivate us to explicitly quantify and manage tail risk in our scheduling algorithm.

Risk-sensitive Scheduling. We adopt Conditional Value at Risk (CVaR) (Rockafellar et al., 2000) to quantify the tail risk of requests. Intuitively, CVaR represents the expected value conditioned on the tail of a distribution. Formally, for a random variable X and confidence level $\alpha \in (0, 1)$:

$$\text{CVaR}_\alpha(X) = \mathbb{E}[X \mid X \geq \text{VaR}_\alpha(X)] \quad (9)$$

where $\text{VaR}_\alpha(X) = \inf\{x : \mathbb{P}(X \leq x) \geq \alpha\}$ denotes the α -quantile of X . Unlike single-point metrics such as P90, CVaR provides a more comprehensive characterization of the tail behavior.

Recall that we model output length using the log-t distribution. In practice, LLM generation is forcibly terminated once the output length reaches the `max_tokens` limit. To account for this, we *truncate* the predicted distribution at $x_{\max} = \text{max_tokens}$ (i.e., $\tilde{X} = \min(X, x_{\max})$). For $X \sim \text{Log-t}(\mu, \sigma, \nu)$, define the partial expectation function:

$$\Psi(y) = \int_{-\infty}^y \exp(\mu + \sigma s) \cdot t_\nu(s) ds, \quad (10)$$

which represents the cumulative contribution to the expectation of X over the region $\{Y \leq y\}$. The truncated expectation and CVaR at confidence level α are then given by (derivations in Appendix A):

$$\mathbb{E}[\tilde{X}] = \Psi(y_{\max}) + x_{\max} \cdot [1 - T_\nu(y_{\max})], \quad (11)$$

$$\text{CVaR}_\alpha[\tilde{X}] = \frac{\Psi(y_{\max}) - \Psi(y_\alpha) + x_{\max} \cdot [1 - T_\nu(y_{\max})]}{1 - \alpha}, \quad (12)$$

²<https://github.com/>

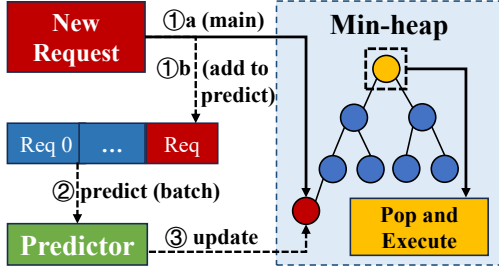


Figure 2. Scheduling workflow for new requests. Solid arrows indicate the main workflow, while dashed arrows represent the asynchronous prediction workflow.

where $y_{\max} = (\ln x_{\max} - \mu)/\sigma$, $y_{\alpha} = T_{\nu}^{-1}(\alpha)$ is the α -quantile, $t_{\nu}(\cdot)$ is the PDF of the standard t-distribution (defined in Eq. (2)), and $T_{\nu}(\cdot)$ denotes its CDF.

For scheduling efficiency, we employ *Monte Carlo sampling* with 10,000 samples to evaluate these metrics instead of numerical integration.

For each incoming request, let X denote the predicted output length distribution, we compute a scheduling score as:

$$\text{Score} = \mathbb{E}[X \mid X \leq x_{\max}] + \beta \cdot \text{CVaR}_{\alpha}[X \mid X \leq x_{\max}] \quad (13)$$

We set $\alpha = 0.9$ to capture the top 10% tail behavior. The parameter β controls risk sensitivity and is *adaptively* adjusted between 0.1 and 0.5 based on the *waiting queue length*. The rationale is that as more requests accumulate in the queue, long requests tend to block a larger number of subsequent requests, thus requiring a more conservative risk control strategy (i.e., a larger β).

Starvation Prevention. A well-known challenge of SJF-based scheduling is that under high load, long requests tend to be starved by short requests, resulting in excessively long waiting times and compromising scheduling fairness. To mitigate this issue, we continuously monitor the waiting time of each request. For every 30 seconds a request waits, we multiply its score by 0.9, and this process continues until the request begins execution. This simple mechanism prevents the starvation of individual requests without disrupting the overall scheduling scheme.

4. Implementation

We also optimize the deployment to improve the efficiency of prediction and scheduling.

Existing prediction-based scheduling methods typically employ *synchronous* prediction (Qiu et al., 2024; Shahout et al., 2025), where incoming requests are first fed into the pre-

dictor, and the prediction results are then used as sorting keys before enqueueing. Since requests are blocked during synchronous prediction, waiting to form a batch would introduce unacceptable latency. As a result, requests are typically predicted one at a time, forgoing the throughput benefits of batching. This synchronous approach is particularly inefficient in the prevalent continuous batching scenario: (1) under low load, new requests should be directly added to the running batch without incurring additional blocking overhead from synchronous prediction; (2) under high load, too many requests accumulate while waiting for prediction, and the unbatched low-throughput prediction cannot promptly identify the shortest requests.

To address these issues, we design an *asynchronous* prediction mechanism along with dynamic batching.

Asynchronous Prediction. We decouple the scheduler into a *main thread* and a *prediction thread*. The prediction thread is responsible for efficiently predicting pending requests without blocking request enqueueing or the main thread’s processing. The main thread, on the other hand, selects a request from the waiting queue to fill a vacant slot in the running batch whenever one becomes available, even if the request has not yet completed or even begun prediction.

Specifically, as illustrated in Figure 2, when the scheduler receives a new request, it inserts the request into the waiting queue (implemented as a *min-heap*) with `max_tokens` as the initial score (1a). Using the maximum tokens as the initial score ensures that unpredicted requests sink to the bottom, allowing already-predicted requests to be executed first and preventing unpredicted long requests from blocking the running batch. Meanwhile, the request is submitted to a prediction queue (1b), where a background thread processes requests *asynchronously* in batches (2). Upon completion of the prediction, the scheduler computes the score and updates the priority of the corresponding request in the heap (3), then re-heapifies to ensure that the minimum element remains at the top.

Under low load, new requests can be directly enqueued and added to the main thread’s processing batch without waiting for the prediction thread to complete.

This design makes it possible to apply batching for higher prediction throughput, thereby enabling an efficient and intelligent scheduling pipeline.

Dynamic Batching. To improve efficiency under varying load conditions, we adopt dynamic batching for the prediction thread. Specifically, requests can wait for a short period to accumulate into a larger batch, and batch prediction is triggered either when a timeout (i.e., 3 ms) expires or when the maximum batch size (i.e., 32) is reached.

Under high load, this batching mechanism allows waiting

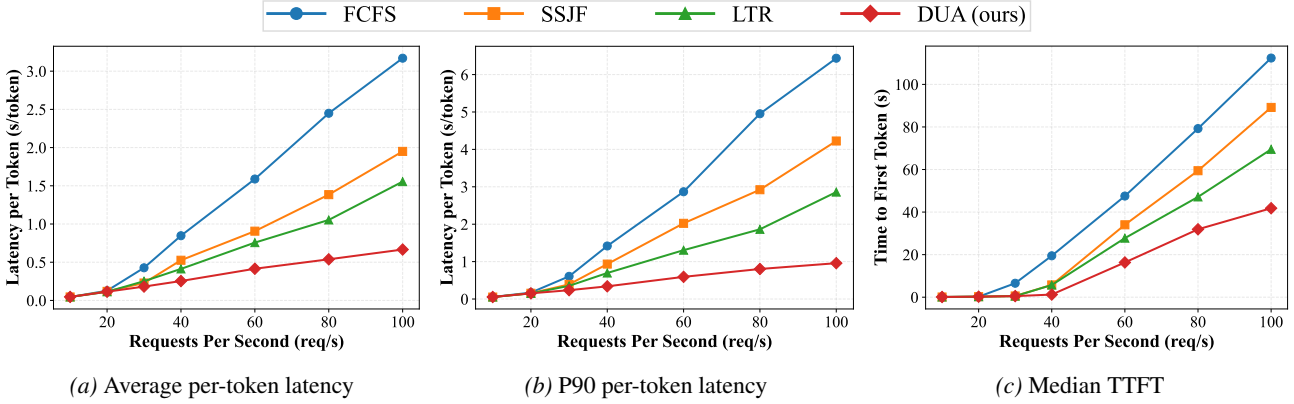


Figure 3. Performance of scheduling strategies for online chatbot serving on the real-world workload with the 8B model.

requests to be predicted rapidly, thereby quickly identifying the shortest requests in a long waiting queue.

According to our experiments, this design reduces the average prediction cost per request from 10.6 ms to 4.3 ms. More importantly, it enables rapid identification of the shortest requests in the waiting queue under high load, further reducing average latency.

Complexity. Retrieving the minimum element takes $O(1)$, while insertion, extraction, and priority updates all require $O(\log n)$. This ensures that the scheduler can efficiently handle requests even under high load.

5. Experiment

In this section, we compare the proposed DUA scheduling strategy against existing state-of-the-art (SOTA) *non-preemptive* methods. Additionally, we conduct parameter tuning and ablation studies to validate our design choices.

5.1. Experimental Setup

Testbed. Our experiments are conducted on a server equipped with 8 NVIDIA A6000 GPUs interconnected via NVLink, 128 virtual CPU cores, and 512 GB of memory.

Serving Models. We evaluate two model scales: Meta-Llama-3-8B-Instruct and Meta-Llama-3-70B-Instruct, both served in FP16 precision. The 70B model is deployed with 8-way tensor parallelism.

Workloads. Our experimental scenarios encompass both online chatbot serving and offline Synthetic Data Generation (SDG) tasks. The test data spans three datasets: LMSYS-Chat-1M (Zheng et al., 2023a), ShareGPT (RyokoAI, 2023), and Alpaca (Taori et al., 2023). LMSYS-Chat-1M and ShareGPT both contain *real* user conversations with LLM-based chatbots. Alpaca is a *synthetic* dataset generated via

self-instruct using GPT-3.5. This setup allows us to assess the performance of each scheduling strategy across diverse scenarios and datasets.

Baselines. We evaluate the following scheduling methods: (1) **FCFS**: First-Come-First-Served, the default scheduling policy in vLLM; (2) **SSJF** (Qiu et al., 2024): A supervised approach that extracts the [CLS] representation of input prompts using a fine-tuned BERT-base model and predicts the *output length* through a two-layer FC network; (3) **LTR** (Fu et al., 2024): A learning-to-rank approach that trains a small OPT model with ListMLE loss to predict the *relative ordering* of request generation lengths (evaluated by Kendall’s Tau); (4) **DUA** (ours): Our proposed method that fits output length distributions using the log-t distribution, extracts semantic features via DeBERTa, fuses information through multi-level pooling to predict *output length distributions*, and performs risk-adaptive scheduling based on waiting queue length.

All scheduling policies are implemented on vLLM 0.11.1.

Training Data. The training data for all scheduling methods are derived from the LMSYS-Chat-1M dataset, with outputs generated by the Meta-Llama-3-8B-Instruct model. Notably, the training data scales are kept *consistent* across methods: SSJF and LTR utilize the first 900K samples, while DUA uses the first 45K samples with 20 repeated generations per sample (tuned in Appendix B) for distribution fitting.

Metrics. For online chatbot serving, we adopt per-token latency and Time-To-First-Token (TTFT) as evaluation metrics, both of which directly reflect user experience: per-token latency characterizes the reading pace during chatbot interactions, while TTFT measures the waiting time before receiving the initial response. For offline SDG tasks, we evaluate the data generation throughput using two metrics: the time required to generate a fixed number of samples and the number of samples generated within a fixed time

Table 3. Generalization performance of scheduling strategies across different datasets and models under a request rate of 100 RPS. Training data for all methods are derived from LMSYS-Chat-1M using the 8B model.

Testing Dataset	Model Size	Average Per-token Latency (s/token)				P90 Per-token Latency (s/token)			
		FCFS	SSJF	LTR	DUA	FCFS	SSJF	LTR	DUA
LMSYS-Chat-1M	70B	9.08	5.50	4.34	2.41	16.13	8.24	7.03	4.05
ShareGPT	8B	1.66	0.95	0.86	0.54	3.27	1.84	1.56	0.88
	70B	4.36	2.43	2.22	1.41	8.78	4.90	3.78	2.22
Alpaca	8B	1.45	0.71	0.83	0.52	3.62	1.47	1.76	0.93
	70B	4.52	2.06	2.36	1.54	11.36	4.32	5.14	2.81
Testing Dataset	Model Size	Average TTFT (s)				P90 TTFT (s)			
		FCFS	SSJF	LTR	DUA	FCFS	SSJF	LTR	DUA
LMSYS-Chat-1M	70B	319.51	273.30	252.20	204.03	618.21	551.15	507.35	475.10
ShareGPT	8B	161.56	132.58	120.03	106.52	316.07	296.69	273.87	257.53
	70B	417.28	342.45	346.50	288.79	807.27	744.59	696.75	660.28
Alpaca	8B	83.65	62.93	65.33	54.18	159.72	140.80	142.17	123.02
	70B	235.64	171.14	174.31	146.14	444.52	395.78	394.10	359.08

budget. Considering the growing diversity of LLMs, we also assessed the generalization of each scheduling strategy.

5.2. Online Chatbot Serving

Figure 4 illustrates the performance of each method under the LMSYS-Chat-1M dataset and the 8B model. As the requests per second (RPS) increases, all metrics exhibit an upward trend, with the default FCFS policy showing the steepest rise. Existing methods (i.e., SSJF and LTR) effectively reduce per-token latency and TTFT by approximately $1.63\times$ and $1.26\times$, respectively, by predicting output lengths and reordering requests accordingly. Our proposed DUA method further improves upon these results by modeling and predicting output length distributions, thereby leveraging the inherent uncertainty of LLMs. As shown in Figure 4d, at 100 RPS, DUA achieves a $4.96\times$ reduction in average per-token latency compared to FCFS, and outperforms SSJF and LTR by $2.93\times$ and $2.33\times$, respectively.

Figure 3c shows the median TTFT metric. At 100 RPS, DUA reduces this metric by $2.69\times$, $2.13\times$, and $1.66\times$ compared to FCFS, SSJF, and LTR, respectively. Additionally, when RPS exceeds 20, the median TTFT under FCFS begins to increase, indicating that more than *half of the requests* start to wait in the queue. For SSJF and LTR, this threshold is 30 RPS, while DUA does not exhibit this phenomenon until RPS exceeds 40.

Furthermore, to evaluate our design of adaptively adjusting risk sensitivity based on queue length, we measured the increase in average per-token latency when the workload rapidly rises (i.e., the RPS increases from 30 to 100). For

FCFS, SSJF, and LTR, this metric is $7.42\times$, $8.55\times$, and $6.17\times$, respectively; whereas for DUA, it is only $3.68\times$. This demonstrates the effectiveness of DUA’s risk-adaptive scheduling. Recall that as the number of waiting requests grows, long requests cause more severe damage to subsequent requests. DUA mitigates this risk by adaptively increasing the risk sensitivity coefficient, thereby deprioritizing requests that may potentially produce long outputs.

Cross-Model and Cross-Dataset Generalization. We further evaluate the generalization capability of each scheduling strategy across different models and datasets. Note that all scheduling models are trained on data derived from the LMSYS-Chat-1M dataset using the Meta-Llama-3-8B-Instruct model. We test each strategy on Meta-Llama-3-70B-Instruct and on the ShareGPT and Alpaca datasets. The results are summarized in Table 3.

On the 70B model, the improvements of SSJF and LTR over FCFS drop significantly to approximately $1.21\times$, whereas DUA maintains a $3.47\times$ speedup.

On the ShareGPT dataset, all methods exhibit some degradation. Nevertheless, DUA still achieves $2.58\times$ (8B model) and $2.92\times$ (70B model) improvements, significantly outperforming FCFS and existing methods. Similar improvements are also observed on the Alpaca dataset.

These results demonstrate that DUA exhibits better generalization compared to existing methods. This can be attributed to the distribution-aware perspective, which explicitly accounts for inherent uncertainty, combined with risk-sensitive adaptive scheduling. These designs enable DUA to achieve

Table 4. In SDG tasks, comparison of scheduling methods on different training datasets and models, tested on Alpaca dataset.

Training Dataset	Model Size	Time for 3k Samples (s)				#Samples within 3 min			
		FCFS	SSJF	LTR	DUA (ours)	FCFS	SSJF	LTR	DUA (ours)
LMSYS-Chat-1M	8B	229.37	144.93	139.53	98.12	2292	3488	3672	4762
	70B	559.30	324.52	316.98	246.10	861	1557	1895	2524
Alpaca	8B	229.37	135.79	130.67	95.08	2292	3766	3663	4869
	70B	559.30	311.06	314.36	240.19	861	1634	1708	2659

Table 5. Ablation study on distribution family and score computation method. Online chatbot experiments use the LMSYS-Chat-1M dataset, and offline SDG experiments use the Alpaca dataset. All experiments are conducted on the Meta-Llama-3-8B-Instruct model. For score computation, CVaR is calculated at $\alpha = 0.9$, and β denotes the adaptive risk-sensitivity coefficient.

Strategy	Distribution	Score=	Per-token Latency		TTFT		Time for 3k Samples (s)	#Samples within 3 min
			Average	P90	Average	P90		
DUA	log-t ($\nu=3.5$)	$\mathbb{E}[X] + \beta \cdot \text{CVaR}$	0.67	0.96	64.80	161.14	98.12	4762
	log-t (dynamic ν)	$\mathbb{E}[X] + \beta \cdot \text{CVaR}$	0.69	1.02	63.44	162.53	97.70	4709
	log-normal	$\mathbb{E}[X] + \beta \cdot \text{CVaR}$	1.63	3.37	80.48	197.34	142.21	3584
	log-t ($\nu=3.5$)	$\mathbb{E}[X]$	0.75	1.21	70.04	173.78	108.51	4257
	log-t ($\nu=3.5$)	$\mathbb{E}[X] + 0.1 \cdot \text{CVaR}$	0.72	1.15	68.27	169.96	104.76	4391
	log-t ($\nu=3.5$)	$\mathbb{E}[X] + 0.3 \cdot \text{CVaR}$	0.71	1.18	68.71	170.59	105.04	4422

strong performance across diverse experimental settings.

5.3. Offline Synthetic Data Generation

Synthetic Data Generation (SDG) is emerging as an important inference workload for LLMs due to the scarcity of high-quality data (Chan et al., 2024). In SDG tasks, shorter responses are generally preferred due to their better cost-efficiency, greater diversity, and avoidance of verbosity bias that may arise from lengthy outputs (Fu et al., 2024; Singhal et al., 2024; Dubois et al., 2024). Consequently, SJF-like scheduling strategies are inherently well-suited for this application scenario. Since the Alpaca dataset is synthetic data generated via self-instruct, it naturally serves as a suitable benchmark for evaluating SDG scenarios (Taori et al., 2023).

Following prior work (Fu et al., 2024), we submit 10K prompts and employ two metrics to evaluate the performance of different strategies: (1) the time required to generate half of the data, and (2) the number of samples generated within 3 minutes. The results are presented in Table 4.

Results. SJF and LTR outperform FCFS on both metrics, achieving faster generation speeds, while DUA achieves further improvements. In terms of the time to generate 3K samples, DUA achieves speedups of approximately $2.34\times$, $1.48\times$, and $1.42\times$ over FCFS, SSJF, and LTR, respectively. In cross-model and cross-dataset scenarios, although the improvements are somewhat diminished, DUA still maintains strong performance. These results demonstrate the effectiveness of our design: considering the output length of

requests from a distributional and uncertainty perspective.

5.4. Ablation Study

We conduct ablation studies to analyze the contributions of each design choice in DUA.

Distribution Family. As discussed in Section 3.3, we evaluate the fitting performance of several commonly used heavy-tailed distribution families and ultimately select the log-t distribution with a fixed $\nu = 3.5$. We also experiment with other distribution families that achieve $> 50\%$ pass rates in the KS test (Table 2), training models and performing scheduling. Table 5 presents the results of both online and offline experiments. Compared to DUA, the log-normal distribution, which exhibits inferior fitting performance (60.3% vs. 90.6%), leads to degraded scheduling performance. This indicates that fitting quality directly affects the accuracy of distribution modeling and scheduling effectiveness. The log-t distribution with a dynamic ν parameter achieves comparable performance to DUA. Considering model complexity and scheduling efficiency, we adopt a fixed ν parameter.

Score Computation. Equation 13 presents our score computation method, which incorporates the expectation, CVaR, and an adaptive risk-sensitivity coefficient β . We also evaluate alternative score computation methods, as shown in Table 5. Using only the expectation yields reasonable performance. Incorporating CVaR further improves the results, with the most notable gains achieved when the adaptive risk-sensitivity coefficient β is introduced.

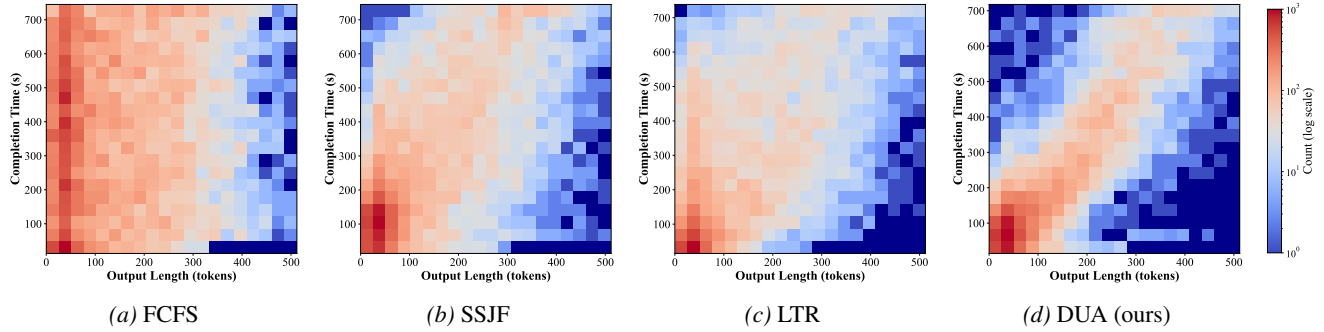


Figure 4. Performance of scheduling strategies...

6. Conclusion

This paper presents DUA, a distribution-aware and uncertainty-aware scheduling policy for LLM inference requests. We employ the log-t distribution to model the heavy-tailed output length distribution of inference requests, and introduce CVaR with risk-adaptive scheduling to handle the risk of requests generating unexpectedly long outputs. Experimental results demonstrate that our method achieves substantial improvements over existing SOTA methods on both online chatbot serving and offline SDG tasks.

Accessibility

Authors are kindly asked to make their submissions as accessible as possible for everyone including people with disabilities and sensory or neurological differences. Tips of how to achieve this and what to pay attention to will be provided on the conference website <http://icml.cc/>.

Software and Data

If a paper is accepted, we strongly encourage the publication of software and data with the camera-ready version of the paper whenever appropriate. This can be done by including a URL in the camera-ready copy. However, **do not** include URLs that reveal your institution or identity in your submission for review. Instead, provide an anonymous URL or upload the material as “Supplementary Material” into the OpenReview reviewing system. Note that reviewers are not required to look at this material when writing their review.

Acknowledgements

Do not include acknowledgements in the initial version of the paper submitted for blind review.

If a paper is accepted, the final camera-ready version can (and usually should) include acknowledgements. Such acknowledgements should be placed at the end of the section, in an unnumbered section that does not count towards the

paper page limit. Typically, this will include thanks to reviewers who gave useful comments, to colleagues who contributed to the ideas, and to funding agencies and corporate sponsors that provided financial support.

Impact Statement

Authors are **required** to include a statement of the potential broader impact of their work, including its ethical aspects and future societal consequences. This statement should be in an unnumbered section at the end of the paper (co-located with Acknowledgements – the two may appear in either order, but both must be before References), and does not count toward the paper page limit. In many cases, where the ethical impacts and expected societal implications are those that are well established when advancing the field of Machine Learning, substantial discussion is not required, and a simple statement such as the following will suffice:

“This paper presents work whose goal is to advance the field of Machine Learning. There are many potential societal consequences of our work, none which we feel must be specifically highlighted here.”

The above statement can be used verbatim in such cases, but we encourage authors to think about whether there is content which does warrant further discussion, as this statement will be apparent if the paper is later flagged for ethics review.

References

- Alasmar, M., Clegg, R., Zakhleniuk, N., and Parisi, G. Internet traffic volumes are not gaussian—they are log-normal: An 18-year longitudinal study with implications for modelling and prediction. *IEEE/ACM Transactions on Networking*, 29(3):1266–1279, 2021.
- Anthropic. Claude, 1 2026. URL <https://claude.ai/>. Large language model.
- Apple. Kv prediction for improved time to first token. Apple Machine Learning Research, 2024.

- URL <https://machinelearning.apple.com/research/kv-prediction>.
- Cassidy, D. T., Hamp, M. J., and Ouyed, R. Pricing european options with a log student's t-distribution: A gosset formula. *Physica A: Statistical Mechanics and its Applications*, 389(24):5736–5748, 2010.
- Chan, Y.-C., Pu, G., Shanker, A., Suresh, P., Jenks, P., Heyer, J., and Denton, S. M. Balancing cost and effectiveness of synthetic data generation strategies for llms. In *NeurIPS 2024 Workshop on Fine-Tuning in Machine Learning (FITML)*, 2024. URL <https://openreview.net/forum?id=hRjFiTxv1v>.
- Chatterji, A., Cunningham, T., Deming, D. J., Hitzig, Z., Ong, C., Shan, C. Y., and Wadman, K. How people use chatgpt. Technical report, National Bureau of Economic Research, 2025.
- D'Agostino, R. B. and Stephens, M. A. Goodness-of-fit techniques, 1986.
- Dubois, Y., Liang, P., and Hashimoto, T. Length-controlled alpacaeval: A simple debiasing of automatic evaluators. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=CybBmzWBX0>.
- Fisher, R. A. *Statistical Methods for Research Workers*. Oliver and Boyd, Edinburgh, 1925.
- Fu, Y., Zhu, S., Su, R., Qiao, A., Stoica, I., and Zhang, H. Efficient llm scheduling by learning to rank. *Advances in Neural Information Processing Systems*, 37:59006–59029, 2024.
- Gim, I., Chen, G., Lee, S.-s., Sarda, N., Khandelwal, A., and Zhong, L. Prompt cache: Modular attention reuse for low-latency inference. *Proceedings of Machine Learning and Systems*, 6:325–338, 2024.
- Google. Gemini, 1 2026. URL <https://gemini.google.com/>. Large language model.
- He, P., Liu, X., Gao, J., and Chen, W. Deberta: Decoding-enhanced bert with disentangled attention. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=XPZiaotutsD>.
- He, P., Gao, J., and Chen, W. DeBERTav3: Improving deberta using ELECTRA-style pre-training with gradient-disentangled embedding sharing. In *The Eleventh International Conference on Learning Representations*, 2023. URL <https://openreview.net/forum?id=sE7-XhLxHA>.
- Heckert, N. A., Filliben, J. J., Croarkin, C. M., Hembree, B., Guthrie, W. F., Tobias, P., and Prinz, J. Handbook 151: Nist/sematech e-handbook of statistical methods. 2002.
- Hugging Face. Text generation inference, 2023. URL <https://github.com/huggingface/text-generation-inference>.
- Jin, Y., Wu, C.-F., Brooks, D., and Wei, G.-Y. s^3 : Increasing gpu utilization during generative inference for higher throughput. *Advances in Neural Information Processing Systems*, 36:18015–18027, 2023.
- Kwon, W., Li, Z., Zhuang, S., Sheng, Y., Zheng, L., Yu, C. H., Gonzalez, J., Zhang, H., and Stoica, I. Efficient memory management for large language model serving with pagedattention. In *Proceedings of the 29th symposium on operating systems principles*, pp. 611–626, 2023a.
- 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*, 2023b.
- Massey Jr, F. J. The kolmogorov-smirnov test for goodness of fit. *Journal of the American statistical Association*, 46 (253):68–78, 1951.
- NVIDIA. Tensorrt-llm: A tensorrt toolbox for large language models, 2023. URL <https://github.com/NVIDIA/TensorRT-LLM>.
- OpenAI. Chatgpt, 1 2026. URL <https://chatgpt.com/>. Large language model.
- Piotrowski, G., Bystroniski, M., Holysz, M., Binkowski, J., Chodak, G., and Kajdanowicz, T. J. When will the tokens end? graph-based forecasting for llms output length. In *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)*, pp. 843–848, 2025.
- Qiu, H., Mao, W., Patke, A., Cui, S., Jha, S., Wang, C., Franke, H., Kalbarczyk, Z. T., Başar, T., and Iyer, R. K. Efficient interactive llm serving with proxy model-based sequence length prediction. In *The 5th International Workshop on Cloud Intelligence / AIOps at ASPLOS 2024*, volume 5, pp. 1–7, San Diego, CA, USA, 2024. Association for Computing Machinery.
- Rockafellar, R. T., Uryasev, S., et al. Optimization of conditional value-at-risk. *Journal of risk*, 2:21–42, 2000.
- RyokoAI. Sharegpt52k. <https://huggingface.co/datasets/RyokoAI/ShareGPT52K>, 2023.

- Shahout, R., eran malach, Liu, C., Jiang, W., Yu, M., and Mitzenmacher, M. DON't STOP ME NOW: EMBEDDING BASED SCHEDULING FOR LLMS. In *The Thirteenth International Conference on Learning Representations*, 2025. URL <https://openreview.net/forum?id=7JhGdZvW4T>.
- Singhal, P., Goyal, T., Xu, J., and Durrett, G. A long way to go: Investigating length correlations in RLHF. In *First Conference on Language Modeling*, 2024. URL <https://openreview.net/forum?id=G8LaO1P0xv>.
- Student. The probable error of a mean. *Biometrika*, pp. 1–25, 1908.
- Taori, R., Gulrajani, I., Zhang, T., Dubois, Y., Li, X., Guestrin, C., Liang, P., and Hashimoto, T. B. Stanford alpaca: An instruction-following llama model. https://github.com/tatsu-lab/stanford_alpaca, 2023.
- Weber, R. R. Scheduling stochastic jobs on parallel machines to minimize makespan or flowtime. In *Applied Probability—Computer Science: The Interface*, pp. 327–344. Springer, 1983.
- Yu, G.-I., Jeong, J. S., Kim, G.-W., Kim, S., and Chun, B.-G. Orca: A distributed serving system for {Transformer-Based} generative models. In *16th USENIX Symposium on Operating Systems Design and Implementation (OSDI 22)*, pp. 521–538, 2022.
- Zheng, L., Chiang, W.-L., Sheng, Y., Li, T., Zhuang, S., Wu, Z., Zhuang, Y., Li, Z., Lin, Z., Xing, E. P., et al. Lmsys-chat-1m: A large-scale real-world llm conversation dataset. *arXiv preprint arXiv:2309.11998*, 2023a.
- Zheng, Z., Ren, X., Xue, F., Luo, Y., Jiang, X., and You, Y. Response length perception and sequence scheduling: An llm-empowered llm inference pipeline. *Advances in Neural Information Processing Systems*, 36:65517–65530, 2023b.

A. Derivations for the Truncated Log- t Distribution

Let $Y \sim t(\nu)$ be a standard t -distributed random variable with ν degrees of freedom. We define the random variable

$$X = \exp(\mu + \sigma Y), \quad (14)$$

which follows a log- t distribution, denoted as $X \sim \text{Log-}t(\mu, \sigma, \nu)$.

The probability density function (PDF) of the standard t distribution is given by:

$$t_\nu(y) = \frac{\Gamma(\frac{\nu+1}{2})}{\sqrt{\nu\pi} \Gamma(\frac{\nu}{2})} \left(1 + \frac{y^2}{\nu}\right)^{-\frac{\nu+1}{2}}, \quad (15)$$

with cumulative distribution function (CDF) denoted by $T_\nu(y)$. By applying the change of variables $x = \exp(\mu + \sigma y)$, we obtain the PDF of the log- t distribution:

$$f(x \mid \mu, \sigma, \nu) = \frac{1}{\sigma x} \cdot t_\nu\left(\frac{\ln x - \mu}{\sigma}\right), \quad x > 0. \quad (16)$$

We define the truncated random variable as $\tilde{X} = \min(X, x_{\max})$, and let

$$y_{\max} = \frac{\ln x_{\max} - \mu}{\sigma}. \quad (17)$$

To simplify subsequent derivations, we introduce the *partial expectation function*:

$$\Psi(y) = \int_{-\infty}^y \exp(\mu + \sigma s) \cdot t_\nu(s) ds. \quad (18)$$

This function admits a natural probabilistic interpretation: $\Psi(y) = \mathbb{E}[X \cdot \mathbf{1}_{\{Y \leq y\}}]$ represents the contribution to the expectation of X from the event $\{Y \leq y\}$. When $y \rightarrow \infty$, we have $\Psi(\infty) = \mathbb{E}[X]$, provided the expectation exists (i.e., when $\nu > 1$ and $\sigma < \nu$).

A.1. Truncated Expectation

The expectation of $\tilde{X}$ can be decomposed into two terms:

$$\mathbb{E}[\tilde{X}] = \underbrace{\int_0^{x_{\max}} x \cdot f(x) dx}_{I_1} + \underbrace{x_{\max} \cdot \mathbb{P}(X > x_{\max})}_{I_2}. \quad (19)$$

Derivation of I_1 . Substituting the PDF and applying the substitution $y = (\ln x - \mu)/\sigma$, we have $x = \exp(\mu + \sigma y)$ and $dx = \sigma \exp(\mu + \sigma y) dy$. The integration limits transform to $y \in (-\infty, y_{\max}]$, yielding:

$$I_1 = \frac{1}{\sigma} \int_0^{x_{\max}} t_\nu\left(\frac{\ln x - \mu}{\sigma}\right) dx = \int_{-\infty}^{y_{\max}} \exp(\mu + \sigma y) \cdot t_\nu(y) dy = \Psi(y_{\max}). \quad (20)$$

Derivation of I_2 . Since $X = \exp(\mu + \sigma Y)$, we have $\mathbb{P}(X \leq x_{\max}) = T_\nu(y_{\max})$, and thus:

$$I_2 = x_{\max} \cdot [1 - T_\nu(y_{\max})]. \quad (21)$$

Combining these results, we obtain:

$$\mathbb{E}[\tilde{X}] = \Psi(y_{\max}) + x_{\max} \cdot [1 - T_\nu(y_{\max})]. \quad (22)$$

This result establishes Eq. (11) in Section 3.

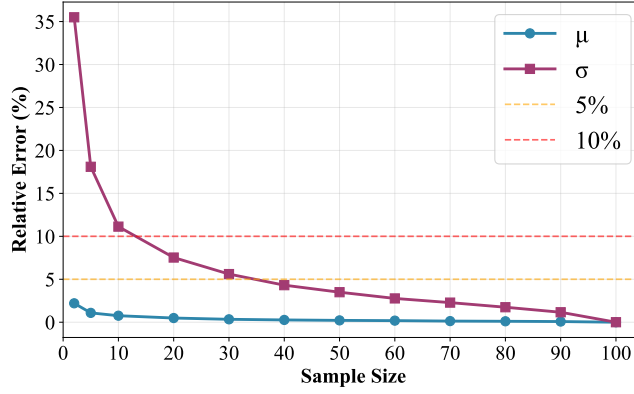


Figure 5. Effect of the number of repetitions on parameter estimation accuracy. The relative error is computed against the baseline obtained with 100 repetitions.

A.2. Truncated Conditional Value-at-Risk

The Conditional Value-at-Risk (CVaR), also known as Expected Shortfall, is defined as the conditional expectation beyond the Value-at-Risk (VaR) threshold:

$$\text{CVaR}_\alpha[X] = \mathbb{E}[X \mid X > \text{VaR}_\alpha[X]], \quad (23)$$

where $\text{VaR}_\alpha[X] = F^{-1}(\alpha)$ denotes the α -quantile.

For the truncated distribution $\tilde{X}$ with confidence level $\alpha < T_\nu(y_{\max})$, the VaR is given by:

$$v_\alpha \triangleq \text{VaR}_\alpha[\tilde{X}] = \exp(\mu + \sigma y_\alpha), \quad \text{where } y_\alpha = T_\nu^{-1}(\alpha). \quad (24)$$

By the definition of conditional expectation:

$$\text{CVaR}_\alpha[\tilde{X}] = \frac{\mathbb{E}[\tilde{X} \cdot \mathbf{1}_{\{\tilde{X} > v_\alpha\}}]}{\mathbb{P}(\tilde{X} > v_\alpha)}. \quad (25)$$

The denominator equals $1 - \alpha$ since $v_\alpha < x_{\max}$. For the numerator, the event $\{\tilde{X} > v_\alpha\}$ comprises two cases: (i) $v_\alpha < X < x_{\max}$, where $\tilde{X} = X$, and (ii) $X \geq x_{\max}$, where $\tilde{X} = x_{\max}$. Thus:

$$\mathbb{E}[\tilde{X} \cdot \mathbf{1}_{\{\tilde{X} > v_\alpha\}}] = \underbrace{\int_{v_\alpha}^{x_{\max}} x \cdot f_X(x) dx}_{J_1} + \underbrace{x_{\max} \cdot \mathbb{P}(X \geq x_{\max})}_{J_2}. \quad (26)$$

Following the same substitution as in Section A.1, we obtain:

$$J_1 = \int_{y_\alpha}^{y_{\max}} \exp(\mu + \sigma y) \cdot t_\nu(y) dy = \Psi(y_{\max}) - \Psi(y_\alpha), \quad J_2 = x_{\max} \cdot [1 - T_\nu(y_{\max})]. \quad (27)$$

Combining these results yields:

$$\text{CVaR}_\alpha[\tilde{X}] = \frac{1}{1 - \alpha} [\Psi(y_{\max}) - \Psi(y_\alpha) + x_{\max} \cdot [1 - T_\nu(y_{\max})]], \quad (28)$$

This result establishes Eq. (12) in Section 3.

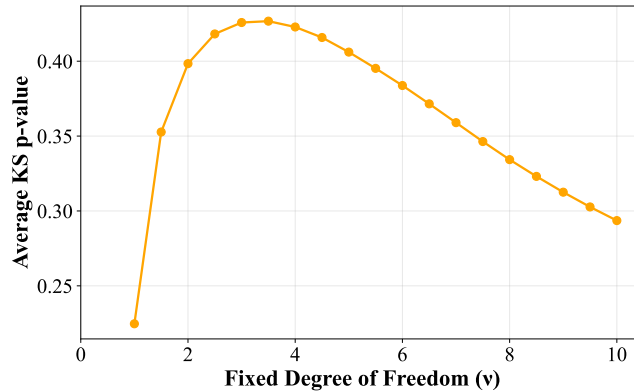


Figure 6. Effect of the degrees of freedom parameter ν on fitting quality.

B. Sensitivity to Sample Size

We sample 1k prompts and establish a baseline by fitting distribution parameters with 100 repetitions. We then vary the number of repetitions and compute the relative error against this baseline. As shown in Figure 5, the μ parameter exhibits little variation, as it controls the location of the distribution and represents the typical case, which can be estimated with only a few samples. In contrast, the error in σ decreases rapidly as the number of repetitions increases, since σ governs the scale of the distribution and requires more samples to estimate accurately.

Although increasing the sample size yields more precise parameter estimates, it also incurs higher data collection costs. We select 20 repetitions as a trade-off between accuracy and efficiency.

C. Sensitivity to Fixed ν value

In practice, we use a fixed degree of freedom parameter ν for fitting. To determine this, we vary ν from 1 to 10, fit the distribution for each value, and compute the average p-value using the Kolmogorov-Smirnov (KS) test. As shown in Figure 6, $\nu = 3.5$ achieves the best performance.

D. You *can* have an appendix here.

You can have as much text here as you want. The main body must be at most 8 pages long. For the final version, one more page can be added. If you want, you can use an appendix like this one.

The `\onecolumn` command above can be kept in place if you prefer a one-column appendix, or can be removed if you prefer a two-column appendix. Apart from this possible change, the style (font size, spacing, margins, page numbering, etc.) should be kept the same as the main body.