Efficient Multi-Turn LLM Serving with Contextual Similarity and Resource-Aware Scheduling

Weizhi Kong
Harbin Institute of Technology, Shenzhen
Shenzhen, China
23s151058@stu.hit.edu.cn

Weizhe Zhang*
Harbin Institute of Technology, Shenzhen
Shenzhen, China
wzzhang@hit.edu.cn

Yiming Wang
China University of Geosciences, Beijing
Beijing, China
yimingw@cugb.edu.cn

***Abstract*—Large Language Models (LLMs) have achieved tremendous success. However, their autoregressive nature results in highly uncertain execution times for user requests, making it difficult to prioritize requests and often causing Head-of-Line (HOL) blocking in inference systems. Existing approaches frequently optimize scheduling by building lightweight predictors for model response lengths. Yet, these predictors have limited context input capacity, and in multi-turn dialogue scenarios, naively concatenating the entire context may lead to truncation and loss of critical information. In this work, we experimentally reveal a correlation between the context of multi-turn dialogues and the lengths of their responses. By extracting key features from the dialogue context, we construct a multi-turn dialogue predictor that achieves more accurate length predictions. Leveraging these predictions, we simulate the request generation process to effectively reduce preemptions caused by insufficient GPU memory. Comparative experiments on a state-of-the-art LLM inference system show that our method reduces the Time to First Token (TTFT) by 47.98% and the average per-token generation latency by 76.56%.**

***Index Terms*—Length Prediction, Scheduling Policy, LLM Serving, Latency Reduction**

I. Introduction

Large Language Models (LLMs) have demonstrated powerful capabilities in multiple domains, including code generation [1], psychological counseling [2], and financial analysis [3]. With the rapid growth in demand for model services, resource competition between different requests has become increasingly intense, leading to significant increases in service latency. Therefore, developing efficient LLM inference scheduling strategies to improve resource utilization and reduce response latency for user requests is crucial for ensuring service quality and system efficiency.

Unlike traditional feed-forward DNN models (e.g., VGG [4], ResNet [5], GoogLeNet [6]), whose inference latency is largely deterministic given fixed input shapes, hardware, and model configurations [7], LLMs operate differently. LLMs generate tokens autoregressively, where each new token is produced based on the sequence of preceding ones. This process continues until a termination condition is met, with all intermediate results (the KV cache) being stored in GPU memory. Consequently, the total generation time is a product of the number of tokens to be generated and the average time per token. A key challenge, however, is that the total number of output tokens is unknown prior to the completion of the inference task, making scheduling difficult.

To achieve efficient inference services for LLMs, state-of-the-art inference systems such as Orca [8] introduce continuous batching, and vLLM [9] proposes PagedAttention, both of which have become key technologies for efficient LLM inference. However, they assume that the generation length is unknown and adopt a first-come-first-served (FCFS) scheduling strategy. Under high load, the FCFS strategy can lead to Head-of-Line (HOL) blocking, which in turn increases inference service latency. For example, suppose a request with a long response length arrives slightly earlier than a shorter one, with their arrival times being almost identical. Since the system can only schedule one request at a time, the shorter request will still be forced to wait until the longer one finishes, significantly increasing the average job waiting time.

Furthermore, during LLM inference, the KV cache is stored in GPU memory and grows continuously as generation proceeds. For modern LLMs, the KV cache of a single request can occupy several gigabytes of memory. Under high system load, GPU memory capacity often becomes a bottleneck, leading to frequent memory allocation failures. To handle such situations, inference systems typically rely on preemption to reclaim GPU resources. Preemption, however, introduces significant overhead. Preempted requests incur additional waiting time, increasing their Job Completion Time (JCT). Moreover, releasing the KV cache of an in-progress request results in wasted computation and reduced memory utilization. Although some systems mitigate this issue by swapping the KV cache to secondary storage, the associated data transfer overhead further degrades overall system performance.

TABLE I
PROPORTION OF SINGLE AND MULTI-TURN DIALOGUES

Model	Single-turns (%)	Multi-turns(%)
ShareGPT [10]	28	72
WildChat-1M [11]	43	57
Lmsys-Chat-1M [12]	45	56

To address the challenge of unknown response lengths in LLM inference, many existing approaches improve request

scheduling by employing lightweight predictors to estimate the response length of incoming requests. However, the input representations used by these predictors are often limited. In multi-turn dialogue scenarios, simply concatenating the entire conversation history as input can lead to context truncation and the loss of critical information. We analyze real-world dialogue datasets, including WildChat-1M [11], ShareGPT [10], and Lmsys-Chat-1M [12], and observe that multi-turn conversations consistently account for approximately 60% of all requests.

In this paper, we investigate the relationship between the similarity of multi-turn dialogues and their output lengths. We observe a clear correlation between inputs from different turns in the dialogue context and their corresponding responses: higher similarity generally leads to more similar output lengths. Based on this insight, we extracted key features from the multi-turn dialogue context and built a multi-turn dialogue predictor to achieve more accurate predictions. Furthermore, we propose a resource-aware PSJF (RA-PSJF) scheduling strategy, which prioritizes inference requests while explicitly modeling their resource requirements to reduce preemption and improve inference latency. We implement our approach on vLLM and evaluate both prediction accuracy and scheduling performance on two real-world datasets. Compared to the default scheduling policy in vLLM, our method reduces request TTFT and per-token generation latency by 47.98% and 76.56%.

Our contributions are summarized as follows:

- We analyze the relationship between contextual similarity in multi-turn dialogues and response length, and find a clear correlation: turns with higher similarity tend to have more similar output lengths.
- We design a multi-turn dialogue response length predictor by extracting key contextual features, achieving more accurate predictions while avoiding information loss caused by long-context concatenation and truncation.
- We propose a resource-aware PSJF (RA-PSJF) scheduling strategy that leverages predicted response length to reduce preemption and improve request latency.

II. Related Works

A. LLM output length prediction

The earliest work [13] proposes Perception in Advance (PiA), which requires the LLM itself to predict before generating a response. However, this approach has potential impacts on generation quality. Subsequent work introduced Perception Only (PO), decoupling prediction from response generation, but this introduces significant inference overhead. Magnus [14] identifies a linear relationship between input and response length for specific tasks. It is based on embedding vectors, applying instructions and user requests as inputs, and achieving regression prediction through a compressor and random forest regressor. Other methods build classifiers based on transformers [15]. S^3 [16] constructs a 10-class classification model using DistBERT [17] to predict the response length; [18], similar to S^3, uses a BERT [19] to build a classification prediction model, but this approach concatenates historical information from multi-turn dialogues as input to support multi-turn dialogue scenarios. One study employs the Learning to Rank [20] algorithm to train the OPT [21] models for ranking predictions of user requests. However, this design hinders other optimization opportunities in inference, including resource estimation, energy efficiency optimization [22], [23], and job distribution [24].

Existing lightweight predictors are limited in input capacity, which prevents accurate prediction for multi-turn dialogues due to context truncation and information loss. Large prediction models, such as Perception Only (PO), mitigate this issue but incur high resource usage and additional prediction latency, reducing overall efficiency.

B. Prediction-based scheduling

Some studies [14], [18] employ traditional batching methods, which require maintaining all requests until the entire batch's inference is completed. The optimization of these methods aims to improve resource utilization by increasing batch size while ensuring that there are no out-of-memory (OOM) errors. With the gradual maturation of continuous batching, S^3, combined with Orca [25], dynamically evaluates the remaining resources at each iteration and prioritizes requests based on their resource demand size, with the optimization goal of maximizing GPU memory utilization. DynamoLLM [23], on the other hand, groups requests based on predicted lengths, distributes them to different LLM inference instances, and achieves energy efficiency optimization by adjusting GPU frequencies accordingly. Other works [18], [26] focus on reducing job waiting time and latency by employing predicted shortest job first (PSJF) or predicted shortest remaining job first (PSRJF) scheduling strategies. However, these methods do not fully consider the optimization potential of prediction results for job preemption.

III. Methodology

In this section, we first introduce the scheduling system workflow. Subsequently, we detail the analysis of multi-turn dialogue context relationships, as well as the structure of the constructed multi-turn dialogue predictor and the extraction of input features. Finally, we introduce the implementation of the scheduling strategy.

A. System Architecture

As shown in Fig. 1, when a user initiates a request, we first select the corresponding predictor to forecast the response length based on whether the request is a multi-turn dialogue. Subsequently, the request with the predicted length is sent to the user request pool, which dynamically maintains a waiting queue using the prediction results as priority. Then, during the iterative scheduling process, when scheduling requests from the waiting queue, we evaluate the risk of preemption caused by the pending request based on the current KV cache redundancy and the status of jobs in the running queue, and

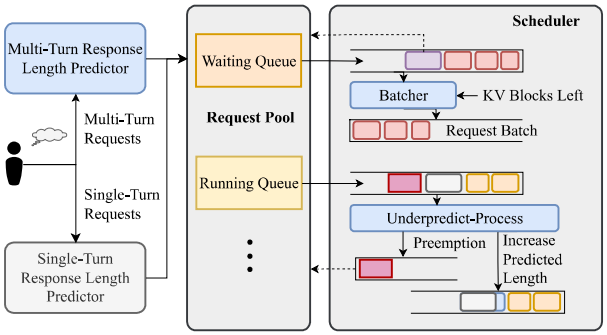


Fig. 1. Workflow of RA-PSJF with response length prediction.

decide whether to schedule the request. For running requests, if a request exceeds its predicted length, we allocate additional budget and re-evaluate it to avoid immediate preemption, thereby reducing the average request latency.

B. Predictor Design

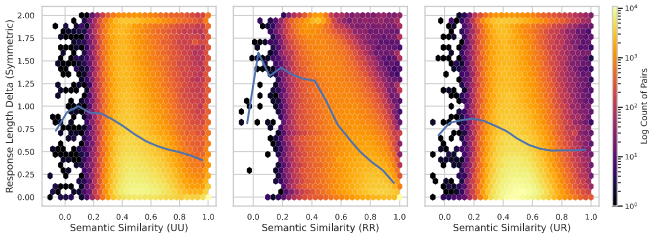
In multi-turn dialogue scenarios with long inputs, we need to extract informative features from the dialogue context to preserve relevant information under limited input capacity. GPTCache [27] observed that semantically similar inputs tend to produce similar outputs. Additionally, BurstGPT [28] reports a linear relationship between input and output lengths. Motivated by these findings, we investigate whether contextual similarity in multi-turn dialogues correlates with the response length of each turn. To evaluate this hypothesis, we analyze multi-turn dialogue datasets from WildChat-1M and Lmsys-Chat-1M. Our primary objective is to examine how the semantic consistency of dialogue history influences the variation in subsequent response lengths.

We utilize an Embedding model as an encoder to map the input and response of each turn into a high-dimensional vector space. We randomly sampled 10^5 sets of multi-turn dialogues from the dataset and mapped the dialogues into a shared vector space. Let u_t and r_t denote the user input and model response at the t-th turn, respectively. We use the Cosine Similarity between vectors to calculate the semantic similarity S_{sim}. To measure the fluctuation of response length, we introduce the Symmetric Relative Length Difference (SRLD). Compared to standard absolute difference, this metric normalizes the deviation, thereby mitigating bias caused by different baseline lengths (e.g., a difference of 10 tokens is significant for a short response but negligible for a long one). For any two turns t and j (where $j < t$), the length difference $\delta_{t,j}$ is defined as:

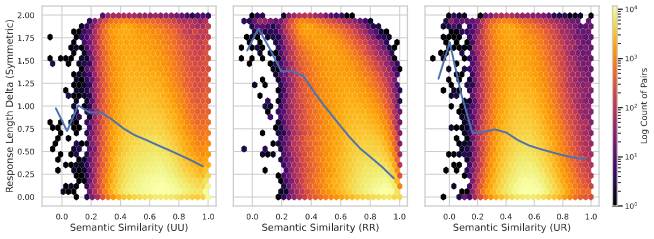
$$\delta_{t,j} = \frac{|L_t - L_j|}{0.5 \times (L_t + L_j)} \tag{1}$$

where L represents the token length of the response. This metric restricts the value range to $[0, 2]$, ensuring numerical stability for statistical analysis.

We investigated the impact of semantic similarity on length variation from three different dimensions. As shown in Fig. 2,



(a) Lmsys-Chat-1M



(b) WildChat-1M

Fig. 2. Average expected response length deviation under different semantic similarity levels.

to handle high-density data, we used Hexbin plots to visualize the distribution of paired data points (S_{sim}, δ). (1) **Input Consistency (User-User, u_t vs. $u_{<t}$)**: We measure the similarity between the current user query and historical user queries. This dimension captures whether users repeating similar questions or rephrasing queries leads to predictable response lengths. (2) **Output Self-Consistency (Model-Model, r_t vs. $r_{<t}$)**: We compare the current response with historical model outputs. High similarity here indicates the model is reiterating previous information, intuitively suggesting length stability. (3) **Response-Input Dependency (User-Model, u_t vs. $r_{<t}$)**: We analyze the relationship between the current user input and previous model responses. This captures scenarios such as "follow-up questions" or users quoting model content, aiming to explore whether high semantic overlap with historical content determines the length of newly generated content.

Fig. 2a and Fig. 2b illustrate the average expected response length deviation under different semantic similarity levels. Although LLM generation involves inherent randomness, a clear linear relationship is observed: higher semantic similarity corresponds to smaller differences in response lengths. This trend is particularly evident in Input Consistency and Output Self-Consistency.

Motivated by the observed correlation between contextual similarity and response length, we extract feature inputs for multi-turn dialogues. Specifically, for each historical turn, we compute similarity and length information from the dialogue context to form a sequential representation, instead of naively concatenating the entire context as input to the predictor.

The model architecture is illustrated in Fig. 3. The predictor first employs a frozen embedding model $E(\cdot)$ to vectorize dialogue texts and compute semantic similarity scores. These features are then processed by an LSTM network [29], which

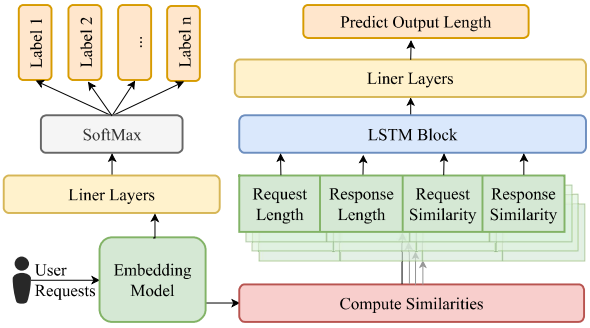


Fig. 3. The architecture of the response length predictor, with the left side designed for single-turn dialogue prediction and the right for multi-turn dialogue prediction.

is well-suited for sequential modeling. By capturing temporal patterns in the constructed feature sequence, the LSTM can effectively model the evolution of contextual similarity and response length trends across dialogue turns. Finally, three linear layers map the extracted features to predicted lengths.

Formally, for requests with dialogue history $\mathcal{H}_i = \{(x_k^{\text{in}}, x_k^{\text{out}})\}_{k=1}^{t-1}$, we compute semantic relevance between the current request and historical turns as

$$s_k^{\text{in}} = \cos\left(E(x_t^{\text{in}}), E(x_k^{\text{in}})\right), \tag{2}$$

$$s_k^{\text{out}} = \cos\left(E(x_t^{\text{in}}), E(x_k^{\text{out}})\right), \tag{3}$$

where $\cos(\cdot,\cdot)$ denotes cosine similarity. Here, x_t^{in} is the current input, while x_k^{in} and x_k^{out} denote the k-th historical request and response, respectively.

We then construct a temporal sequence of dialogue features

$$\mathcal{S}_i = \left\{\left(|x_k^{\text{in}}|, |x_k^{\text{out}}|, s_k^{\text{in}}, s_k^{\text{out}}\right)\right\}_{k=1}^{t-1}, \tag{4}$$

where $|\cdot|$ denotes sequence length. This sequence integrates dialogue length statistics and semantic dependencies, and is processed by an LSTM:

$$\mathbf{h}_k, \mathbf{c}_k = \text{LSTM}(\mathbf{h}_{k-1}, \mathbf{c}_{k-1}; \mathbf{z}_k), \quad \mathbf{z}_k \in \mathcal{S}_i, \tag{5}$$

where $\mathbf{h}_k$ and $\mathbf{c}_k$ are the hidden and cell states at step k, respectively. The final hidden state $\mathbf{h}_{t-1}$ is passed through a regression layer to yield the predicted response length $\hat{L}_i$. For the embedding model, we adopt BGE-M3 [30] and limit the input sequence length to a maximum of 512 tokens per turn. The model is trained using the Adam optimizer with the L1 loss.

For single-turn dialogues, where no historical information is available, we directly encode the current input x_t^{in} using the frozen embedding model $E(\cdot)$ and feed the resulting embedding to a linear classification head that outputs a discrete length category. Unlike prior work, we do not update the parameters of the embedding model. Experiments (Section IV-B) demonstrate that semantic similarity alone is sufficient for coarse-grained prediction. The single-turn predictor is trained using the Adam optimizer with the Negative Log-Likelihood (NLL) loss.

Algorithm 1: Resource-Aware Scheduling Algorithm

Input: Running set $\mathcal{R}$, Waiting set $\mathcal{W}$, Search depth D
Output: Next execution batch $\mathcal{B}$

```
// Initialize with running requests
B ← R;   K_free ← getFreeKV()
Sort W by L̂_i ascending;   n_fail ← 0
foreach req ∈ W while n_fail < D do
    L_in ← promptLen(req)
    if L_in > K_free then
        n_fail ← n_fail + 1; continue
    end
    K_sim ← K_free − L_in;   Safe ← True
    E ← sortByFinishTime(B ∪ {req})
    t_prev ← 0
    foreach r_j ∈ E do
        Δt ← L̂_{r_j} − t_prev
        N_active ← |E| − index(r_j)
        if N_active · Δt > K_sim then
            Safe ← False; break
        end
        // Release finished req memory
        K_sim ← K_sim − (N_active · Δt) + mem(r_j)
        t_prev ← L̂_{r_j}
    end
    if Safe then
        B ← B ∪ {req}
        K_free ← K_free − L_in
    else
        n_fail ← n_fail + 1
    end
end
return B
```

C. Scheduler Policy

In continuous batching inference such as vLLM [31] and Orca [25], each iteration allows the scheduler to schedule requests from the request pool. Based on predicted response lengths $\hat{L}_i$, we prioritize shorter requests, which reduces waiting time and mitigates HOL blocking. Prior works have optimized scheduling using a PSJF policy, but they do not consider how predicted lengths affect preemption. When GPU memory is insufficient, the inference system will preempt requests and reschedule them once resources are available. This preemption not only wastes GPU memory but also delays requests that could have been executed. Since resource usage depends on both input and output lengths, we can evaluate candidate requests during scheduling and select only those that do not trigger preemption.

To avoid preemption, we simulate a candidate request r_i with input length L_i^{in} and predicted length $\hat{L}_i$ over $[0, T]$, where $T = \hat{L}_i$. Let K_{total} denote the maximum KV cache capacity, and K_{used} the tokens already occupied in the current iteration. At each simulated time $t \in [1, T]$, the memory safety

TABLE II
PERFORMANCE OF DIFFERENT PREDICTORS ON **WC** (WILDCHAT-1M) AND **LC** (LMSYS-CHAT-1M) MT DATASETS

DS	Method	Class	Acc (%) ↑				MAE ↓
			@32	@64	@128	@256	
WC	S3	10	6.09	12.33	24.81	50.09	277.93
	S3	400	28.27	43.78	62.67	80.37	169.98
	PO-7B	–	14.92	24.61	40.38	61.71	282.53
	PO-14B	–	12.40	20.66	33.88	55.92	312.08
	OUR	–	**32.80**	**52.92**	**74.18**	**90.28**	**105.61**
LC	S3	5	5.13	10.38	21.58	48.19	263.79
	S3	200	42.09	56.35	69.45	82.43	137.90
	PO-7B	–	23.70	40.04	64.43	87.06	151.18
	PO-14B	–	18.09	32.64	54.27	82.34	169.91
	OUR	–	**42.40**	**58.56**	**77.16**	**93.14**	**82.42**

TABLE III
PERFORMANCE OF DIFFERENT PREDICTORS ON **WC** (WILDCHAT-1M) AND **LC** (LMSYS-CHAT-1M) ST DATASETS

DS	Method	Class	Acc (%) ↑				MAE ↓
			@32	@64	@128	@256	
WC	S3	10	5.45	10.86	21.44	42.75	302.44
	S3	400	29.46	43.85	62.47	80.56	170.43
	PO-7B	–	19.71	32.05	49.96	68.88	244.28
	PO-14B	–	12.90	20.95	30.91	46.68	367.96
	OUR	10	4.94	10.21	21.44	45.99	**271.92**
	OUR	400	30.80	45.95	66.09	85.31	**144.85**
LC	S3	5	5.32	10.72	21.57	43.37	260.77
	S3	200	52.24	68.19	83.40	94.05	71.34
	PO-7B	–	26.98	41.70	60.51	80.66	183.85
	PO-14B	–	23.61	35.84	51.86	74.24	210.95
	OUR	5	10.02	20.80	45.78	93.52	**137.29**
	OUR	200	42.04	58.26	77.43	85.31	**81.63**

condition is checked as:

$$K_{\text{used}} + L_i^{\text{in}} + t \cdot A_t \;\le\; K_{\text{total}}, \tag{6}$$

where A_t denotes the number of active requests at time t.

Upon completion of a request r_j at time t, its resource release is simulated by reducing K_{used} accordingly and decrementing A_t. These updates occur only at request completion events, rather than at every token step. The candidate request r_i is scheduled only when the safety condition holds for all $t \in [1, T]$.

To avoid excessive simulation overhead, especially when KV cache is tight and many candidate requests are likely to fail the safety check, we introduce a tunable parameter `SearchDepth` that limits the number of consecutive scheduling failures allowed in one iteration. Once this limit is reached, the scheduler stops admitting new requests, allocates resources to the selected batch, and proceeds to inference execution.

For requests where the predicted length $\hat{L}_i$ exceeds the actual length, r_i finishes earlier and releases resources without preemption. In cases where $\hat{L}_i$ underestimates the true length, the prediction is extended by a fixed margin Δ, `TokenADD`, yielding a new estimate $\hat{L}_i' = \hat{L}_i + \Delta$, and the safety condition is re-evaluated. The request proceeds with the updated budget if the extended simulation succeeds; otherwise, r_i is preempted and its resources reclaimed.

IV. Experiments

A. Experiments Setup

Testbed The experimental evaluation was implemented on a server equipped with two NVIDIA A100 40 GB GPUs and an Intel Xeon Gold 6248R processor with 72 GB of host memory.

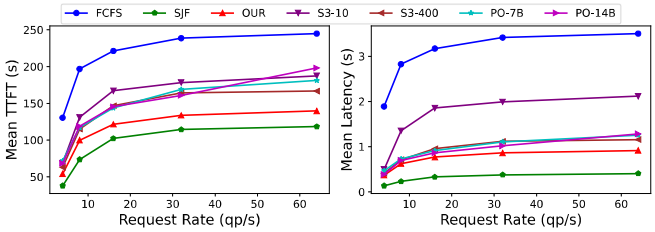
Workloads We selected subsets of conversation data from ChatGPT-3.5 in WildChat-1M and Vicuna-13b in Lmsys-Chat. The context length for ChatGPT-3.5 [32] is set to 4096, while for Vicuna-13b [33] it is set to 2048. From the test set used for training the predictor, we selected 1k dialogue samples to evaluate model serving performance.

Serving Model We use the Qwen2.5-14B-Instruct [34] model for inference on a single GPU, and Qwen2.5-32B-Instruct for inference on two GPUs with tensor parallelism (TP=2). Recomputation is enabled to preempt requests when the KV cache is insufficient for new token allocation.

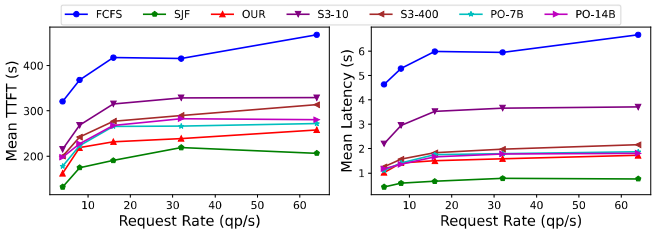
Benchmark and Metrics We evaluate LLM serving performance using mean TTFT (Time to First Token) and mean per-token generation latency. For prediction accuracy, we measure MAE (Mean Absolute Error) and report the proportion of requests with MAE below thresholds of 32, 64, 128, and 256 (ACC@32, 64, 128, 256).

Baselines. We implement our approach on vLLM (v0.42) while maintaining framework flexibility, enabling easy adaptation to other inference systems. We compare it against the following baselines:

- **FCFS**: Default scheduling/batching algorithm in vLLM.
- **S3** [16]: This predictor follows the same architecture as S^3, using DistilBERT as the base model, but incorporates context concatenation for multi-turn dialogues [18]. It employs a bucketing strategy with bucket size $= \frac{\text{max context length}}{\text{number of buckets}}$ and schedules requests using the PSJF algorithm. The maximum input length is limited to 512 tokens, and both S3 and our predictor are trained using up to 10 epochs.
- **PO** [13]: Uses Qwen2.5 models (7B/14B) as dedicated predictors deployed separately with FCFS scheduling. Each request includes a special instruction forcing the model to first output its estimated word count (limited to 15 tokens) before generating the response. The inference stage then performs PSJF scheduling based on these predictions.
- **SJF**: An oracle baseline that schedules requests using the Shortest Job First policy, where the ground-truth response length of each request is used as the predicted value.



(a) Performance on Qwen2.5-14B-Instruct.



(b) Performance on Qwen2.5-32B-Instruct (TP=2).

Fig. 4. Mean TTFT and mean latency of different methods on the WildChat-1M MT dataset.

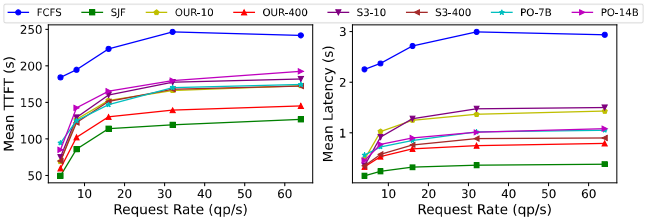


Fig. 5. The mean TTFT and mean latency of different methods in the MT+ST dataset.

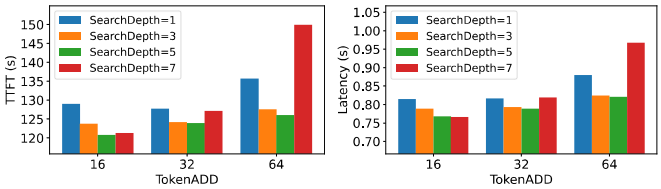


Fig. 6. The impact of `TokenADD` and `SearchDepth` on TTFT and latency, where the request arrival rate is 16 qp/s.

B. Predictions Accuracy

We evaluate the performance of different predictors on multi-turn (MT) and single-turn (ST) datasets, with the results summarized in Table II and Table III (where "Class" is a column name, and its values indicate the number of classification labels in the classification model).

On the MT dataset from WildChat-1M, our predictor achieves the best performance, with an MAE of 105.61 and an ACC@256 of 90.28%. This improvement stems from explicitly modeling contextual similarity and historical length information. In contrast, S3 is limited by coarse bucket granularity and context truncation. PO performs poorly in multi-turn scenarios due to the absence of explicit instruction-turn information in its prediction input. Moreover, regardless of prediction accuracy, deploying an additional LLM as a predictor introduces substantial overhead and latency, making it impractical for real-world serving systems.

On the ST dataset, our embedding-based predictor outperforms the BERT-based classifier used in S3. Specifically, it improves ACC@256 by 3.24% with ten length categories and reduces the prediction error by 30 tokens when using 400 categories, demonstrating the robustness of our design.

For the MT dataset on Lmsys-Chat-1M, although dialogue contexts are generally shorter and truncation is less severe, our predictor still achieves a lower MAE of 82.42. On the corresponding ST dataset, the embedding-based predictor slightly underperforms S3, which we attribute to freezing the embedding model parameters during training.

C. LLM Serving Performance

We select 1k requests from the WildChat-1M MT test set and evaluate different methods under various model settings, with requests issued at different arrival rates. Our approach operates at the scheduler level and can be easily migrated to different inference systems, regardless of the served model or parallelism. Consequently, similar results were achieved across different configurations.

The experimental results in Fig. 4 show that our solution achieves the best performance under multi-turn dialogue workloads. This improvement comes from the predictor's lower prediction errors and the scheduling strategy's ability to avoid unnecessary preemption, which together enable more accurate request prioritization. In contrast, S3 partially mitigates HOL blocking but cannot differentiate priorities within the same coarse bucket, while PO suffers from substantial prediction errors. Compared with the default FCFS in vLLM, our method achieves substantial improvements in both TTFT and per-token generation latency. Specifically, on Qwen2.5-14B-Instruct, TTFT is reduced by an average of 47.98% and per-token generation latency by 76.56%; on Qwen2.5-32B-Instruct, TTFT is reduced by 44.29% and per-token generation latency by 74.70%. Our approach consistently outperforms other prediction-based methods. For reference, the oracle SJF, which uses ground-truth response lengths, provides an upper bound on achievable performance.

D. Mixed Workloads

We further evaluate TTFT and latency on mixed ST and MT workloads, with a ratio of 4:6 as referenced in Table I, as shown in Fig. 5. Our scheduling algorithm is highly sensitive to prediction accuracy. When using a coarse 10-class ST predictor, the performance is suboptimal, whereas increasing the granularity to 400 classes restores results comparable to SJF. Compared to FCFS, our method reduces TTFT by 37.98% and per-token generation latency by 58.91%. The performance of other baseline methods remains consistent with their behavior on MT datasets, and our approach continues to outperform all other prediction-based methods as well as FCFS.

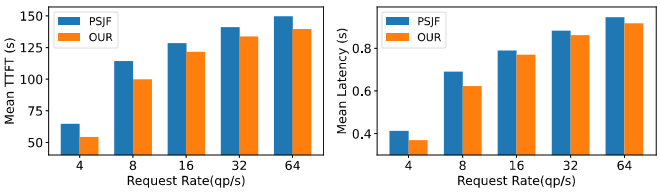


Fig. 7. The optimization effect of our method compared to using only PSJF scheduling, where `TokenADD`=16 and `SearchDepth`=3.

E. Impact of TokenADD and SearchDepth

We evaluate the impact of `SearchDepth` and `TokenADD` on system performance under an arrival rate of 16. As shown in Fig. 6, the optimal range for `SearchDepth` lies between 3 and 5. Smaller values delay scheduling decisions and reduce GPU memory utilization, whereas larger values introduce excessive scheduling delays.

For `TokenADD`, a value of 16 yields the best performance, which is consistent with the default vLLM block size. Larger `TokenADD` values increase the risk of preemption for over-predicted requests, leading to inefficient resource usage. We further evaluate these parameters under different arrival rates, and the results exhibit consistent trends in service quality.

F. Ablation Study on Scheduling Strategies

Fig. 7 compares the performance of our scheduler against the PSJF algorithm. Experimental results show that our method outperforms PSJF consistently. In the tested set of 1k requests, our approach reduces TTFT by up to 16.24% and per-token generation latency by 10.46% compared to PSJF. The main reason is that PSJF only considers whether resources can be allocated for new tokens at the current moment, ignoring future resource requirements. When resources are scarce, preempted requests waste the memory they occupy, leading to increased overhead and latency. In contrast, RA-PSJF evaluates both currently available memory and anticipated future resource demands. If a request cannot be scheduled immediately, we prioritize lower-input-length requests even if their predicted response lengths are relatively long. Although preemptions may still occur when predicted lengths are shorter than actual, our approach significantly reduces additional overhead and request latency compared to PSJF.

G. Predictor Overhead

We evaluate the overhead of predictors on 1k requests with a batch size of 1. The average latency of the single-turn predictor is 23.16 ms. For multi-turn dialogues, the latency increases to 61.74 ms due to additional feature extraction and LSTM processing. This overhead remains negligible, as it is still lower than the generation latency of a single token.

V. Conclusion

This paper reveals that in multi-turn dialogues, turns with higher contextual similarity tend to have closer response lengths. Based on this, we design a response length predictor that leverages history length and similarity features to construct sequential inputs, avoiding long-context truncation and improving prediction accuracy. Using these predictions, our RA-PSJF scheduling strategy reduces unnecessary preemptions, further reducing latency. Compared with a BERT-based classifier, prediction error is reduced by 37.9% on WildChat-1M, and compared with vLLM-FCFS, RA-PSJF lowers average per-token latency and TTFT by 76.56% and 47.98%, respectively. Future work will explore integrating vector databases to unify prediction across single and multi-turn dialogues.

Acknowledgment

This work was supported by the Joint Funds of the National Natural Science Foundation of China (Grant No. U22A2036), the National Natural Science Foundation of China (NSFC) / Research Grants Council (RGC) Collaborative Research Scheme (Grant No. 62461160332 & CRS_HKUST602/24), Shenzhen Science and Technology Program (Grant No. ZDCY20250901101035012), the Shenzhen Colleges and Universities Stable Support Program Nos. GXWD20231130110352002, and GXWD20231129102636001, the National Natural Science Foundation of China (Grant No. 62402141), the Guangdong Basic and Applied Basic Research Foundation (2025A1515011785, 2023A1515110271), the open research fund of Pengcheng Laboratory (No.2025KF1B0020), the Open Research Fund of Pengcheng Laboratory (No. 2025KF1A0020), 2025 Research and Development of Security Detection and Defense Software for AI Models project from China Unicom Intelligent Manufacturing Technology & Industry (Guangdong) Co., Ltd. (No. YGS25AE1000001), 2025 Research and Development of Training-Inference Collaborative Management System Based on Domestic Computing Power Resource Pooling project from China Unicom Intelligent Manufacturing Technology & Industry (Guangdong) Co., Ltd. (No. YGS25AE1000002).

References

[1] B. Hui, J. Yang, Z. Cui, J. Yang, D. Liu, L. Zhang, T. Liu, J. Zhang, B. Yu, K. Dang, A. Yang, R. Men, F. Huang, X. Ren, X. Ren, J. Zhou, and J. Lin, "Qwen2.5-coder technical report," *CoRR*, vol. abs/2409.12186, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2409.12186

[2] D. Xue, J. Tu, M. Wang, X. Yan, F. Liu, and J. Hu, "Towards privacy-preserving mental health support with large language models," 2026. [Online]. Available: https://arxiv.org/abs/2601.01993

[3] N. Wang, H. Yang, and C. D. Wang, "Fingpt: Instruction tuning benchmark for open-source large language models in financial datasets," *NeurIPS Workshop on Instruction Tuning and Instruction Following*, 2023.

[4] K. Simonyan and A. Zisserman, "Very deep convolutional networks for large-scale image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1409.1556

[5] K. He, X. Zhang, S. Ren, and J. Sun, "Deep residual learning for image recognition," 2015. [Online]. Available: https://arxiv.org/abs/1512.03385

[6] C. Szegedy, W. Liu, Y. Jia, P. Sermanet, S. E. Reed, D. Anguelov, D. Erhan, V. Vanhoucke, and A. Rabinovich, "Going deeper with convolutions," in *IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2015, Boston, MA, USA, June 7-12, 2015*. IEEE Computer Society, 2015, pp. 1–9. [Online]. Available: https://doi.org/10.1109/CVPR.2015.7298594

[7] L. L. Zhang, S. Han, J. Wei, N. Zheng, T. Cao, Y. Yang, and Y. Liu, "Nn-meter: Towards accurate latency prediction of deep-learning model inference on diverse edge devices," in *Proceedings of the 19th Annual International Conference on Mobile Systems, Applications, and Services*, 2021, pp. 81–93.

[8] B. Wu, Y. Zhong, Z. Zhang, G. Huang, X. Liu, and X. Jin, "Fast distributed inference serving for large language models," *CoRR*, vol. abs/2305.05920, 2023. [Online]. Available: https://doi.org/10.48550/arXiv.2305.05920

[9] W. Kwon, Z. Li, S. Zhuang, Y. Sheng, L. Zheng, C. H. Yu, J. Gonzalez, H. Zhang, and I. Stoica, "Efficient memory management for large language model serving with pagedattention," in *Proceedings of the 29th Symposium on Operating Systems Principles, SOSP 2023, Koblenz, Germany, October 23-26, 2023*, J. Flinn, M. I. Seltzer, P. Druschel, A. Kaufmann, and J. Mace, Eds. ACM, 2023, pp. 611–626. [Online]. Available: https://doi.org/10.1145/3600006.3613165

[10] S. Team, https://sharegpt.com/, 2023.

[11] W. Zhao, X. Ren, J. Hessel, C. Cardie, Y. Choi, and Y. Deng, "Wildchat: 1m chatGPT interaction logs in the wild," in *The Twelfth International Conference on Learning Representations*, 2024. [Online]. Available: https://openreview.net/forum?id=Bl8u7ZRlbM

[12] L. Zheng, W. Chiang, Y. Sheng *et al.*, "Lmsys-chat-1m: A large-scale real-world LLM conversation dataset," in *The Twelfth International Conference on Learning Representations, ICLR 2024, Vienna, Austria, May 7-11, 2024*. OpenReview.net, 2024. [Online]. Available: https://openreview.net/forum?id=BOfDKxfwt0

[13] Z. Zheng, X. Ren, F. Xue, Y. Luo, X. Jiang, and Y. You, "Response length perception and sequence scheduling: An llm-empowered LLM inference pipeline," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/ce7ff3405c782f761fac7f849b41ae9a-Abstract-Conference.html

[14] K. Cheng, W. Hu, Z. Wang, P. Du, J. Li, and S. Zhang, "Enabling efficient batch serving for lmaas via generation length prediction," in *IEEE International Conference on Web Services, ICWS 2024, Shenzhen, China, July 7-13, 2024*. IEEE, 2024, pp. 853–864. [Online]. Available: https://doi.org/10.1109/ICWS62655.2024.00104

[15] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, I. Guyon, U. von Luxburg, S. Bengio, H. M. Wallach, R. Fergus, S. V. N. Vishwanathan, and R. Garnett, Eds., 2017, pp. 5998–6008. [Online]. Available: https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html

[16] Y. Jin, C. Wu, D. Brooks, and G. Wei, "S^3: Increasing GPU utilization during generative inference for higher throughput," in *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine, Eds., 2023. [Online]. Available: http://papers.nips.cc/paper_files/paper/2023/hash/3a13be0c5dae69e0f08065f113fb10b8-Abstract-Conference.html

[17] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of BERT: smaller, faster, cheaper and lighter," *CoRR*, vol. abs/1910.01108, 2019. [Online]. Available: http://arxiv.org/abs/1910.01108

[18] H. Qiu, W. Mao, A. Patke, S. Cui, S. Jha, C. Wang, H. Franke, Z. T. Kalbarczyk, T. Başar, and R. K. Iyer, "Efficient interactive llm serving with proxy model-based sequence length prediction," in *The 5th International Workshop on Cloud Intelligence / AIOps at ASPLOS 2024*, vol. 5. San Diego, CA, USA: Association for Computing Machinery, 2024, pp. 1–7.

[19] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, J. Burstein, C. Doran, and T. Solorio, Eds. Association for Computational Linguistics, 2019, pp. 4171–4186. [Online]. Available: https://doi.org/10.18653/v1/n19-1423

[20] T.-Y. Liu *et al.*, "Learning to rank for information retrieval," *Foundations and Trends® in Information Retrieval*, vol. 3, no. 3, pp. 225–331, 2009.

[21] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. T. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "OPT: open pre-trained transformer language models," *CoRR*, vol. abs/2205.01068, 2022. [Online]. Available: https://doi.org/10.48550/arXiv.2205.01068

[22] J. Stojkovic, E. Choukse, C. Zhang, Í. Goiri, and J. Torrellas, "Towards greener llms: Bringing energy-efficiency to the forefront of LLM inference," *CoRR*, vol. abs/2403.20306, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2403.20306

[23] J. Stojkovic, C. Zhang, Í. Goiri, J. Torrellas, and E. Choukse, "Dynamollm: Designing LLM inference clusters for performance and energy efficiency," *CoRR*, vol. abs/2408.00741, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2408.00741

[24] C. Hu, H. Huang, L. Xu, X. Chen, J. Xu, S. Chen, H. Feng, C. Wang, S. Wang, Y. Bao, N. Sun, and Y. Shan, "Inference without interference: Disaggregate LLM inference for mixed downstream workloads," *CoRR*, vol. abs/2401.11181, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2401.11181

[25] G. Yu, J. S. Jeong, G. Kim, S. Kim, and B. Chun, "Orca: A distributed serving system for transformer-based generative models," in *16th USENIX Symposium on Operating Systems Design and Implementation, OSDI 2022, Carlsbad, CA, USA, July 11-13, 2022*, M. K. Aguilera and H. Weatherspoon, Eds. USENIX Association, 2022, pp. 521–538. [Online]. Available: https://www.usenix.org/conference/osdi22/presentation/yu

[26] T. Joachims, H. Li, T. Liu, and C. Zhai, "Learning to rank for information retrieval (LR4IR 2007)," *SIGIR Forum*, vol. 41, no. 2, pp. 58–62, 2007. [Online]. Available: https://doi.org/10.1145/1328964.1328974

[27] F. Bang, "GPTCache: An open-source semantic cache for LLM applications enabling faster answers and cost savings," in *Proceedings of the 3rd Workshop for Natural Language Processing Open Source Software (NLP-OSS 2023)*, L. Tan, D. Milajevs, G. Chauhan, J. Gwinnup, and E. Rippeth, Eds. Singapore: Association for Computational Linguistics, Dec. 2023, pp. 212–218. [Online]. Available: https://aclanthology.org/2023.nlposs-1.24/

[28] Y. Wang, Y. Chen, Z. Li, X. Kang, Y. Fang, Y. Zhou, Y. Zheng, Z. Tang, X. He, R. Guo, X. Wang, Q. Wang, A. C. Zhou, and X. Chu, "BurstGPT: A real-world workload dataset to optimize llm serving systems," in *Proceedings of the 31st ACM SIGKDD Conference on Knowledge Discovery and Data Mining V.2 (KDD '25)*. Toronto, ON, Canada: ACM, 2025. [Online]. Available: https://doi.org/10.1145/3711896.3737413

[29] S. Hochreiter, "Long short-term memory," *Neural Computation MIT-Press*, 1997.

[30] J. Chen, S. Xiao, P. Zhang, K. Luo, D. Lian, and Z. Liu, "BGE m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation," *CoRR*, vol. abs/2402.03216, 2024. [Online]. Available: https://doi.org/10.48550/arXiv.2402.03216

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

[32] OpenAI, "Introducing chatgpt," https://openai.com/index/chatgpt/, November 2022.

[33] L. Zheng, W.-L. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing, H. Zhang, J. E. Gonzalez, and I. Stoica, "Judging llm-as-a-judge with mt-bench and chatbot arena," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[34] A. Yang, B. Yang, B. Zhang, B. Hui, B. Zheng, B. Yu, C. Li, D. Liu, F. Huang, H. Wei *et al.*, "Qwen2. 5 technical report," *arXiv preprint arXiv:2412.15115*, 2024.